Self-Enforcing Protocols via Co-Utile Reputation Management

Josep Domingo-Ferrer, Oriol Farràs, Sergio Martínez, David Sánchez[1], Jordi Soria-Comas

UNESCO Chair in Data Privacy, Department of Computer Engineering and Mathematics
Universitat Rovira i Virgili, Av. Països Catalans 26, E-43007 Tarragona, Catalonia
{*josep.domingo,oriol.farras,sergio.martinezl,david.sanchez,jordi.soria*}*@urv.cat*

Abstract

Well-designed protocols should be self-enforcing, that is, be such that rational participating agents have no motivation to deviate from them. In addition, protocols can have other interesting properties, such as promoting collaboration between agents in a search for a better outcome. We proposed the notion of *co-utility* [9, 8], which characterizes a situation in which mutual help is the best rational option to take even for purely selfish agents; in particular, if a protocol is co-utile, it is self-enforcing. However, guaranteeing self-enforcement, let alone co-utility, for any type of agent behavior is not possible. To tackle this issue, in this paper we study how reputation mechanisms can be incorporated into existing protocols in order to make them self-enforcing (and optionally co-utile). Moreover, we adapt and extend the well-known EigenTrust reputation calculation mechanism so that: i) it can be applied to a variety of scenarios and heterogeneous reputation needs and, ii) it is itself co-utile, and hence selfish agents are interested in following it. Obtaining a co-utile reputation mechanism creates a "virtuous circle" because: i) the reputation management is self-enforcing and, ii) as a result, it can be used to turn protocols that were not self-enforcing (resp. co-utile) *per se* into self-enforcing (resp. co-utile) ones. Our theoretical contribution is illustrated with a detailed case study focused on co-utile P2P privacy-preserving query submission to a web search engine or a database; formal and empirical validations of this case study are provided.

Keywords: Protocols, Reputation, P2P, Self-enforcement, Co-utility

[1]Corresponding author. Address: Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Av. Països Catalans 26, 43007 Tarragona, Catalonia. Tel.:+34 977 559657; Fax: +34 977 559710. E-mail: david.sanchez@urv.cat

1. Introduction

A protocol can be defined as a sequence of actions prescribed for an interaction between agents, both in the real world (e.g. road traffic rules) and the virtual world (e.g. peer-to-peer computing). A well-designed protocol is such that the participating agents are willing to follow it, that is, they find no motivation to deviate from the protocol prescriptions. In this manner, the protocol can be completed in a self-enforcing way. Ensuring this in the general case (i.e., for any type of agent behavior) is not possible. This is why protocols are usually designed for the most usual type of agents: *rational agents*. A rational –self-interested– agent acts in order to maximize her profit/utility (i.e., the outcome she obtains, such as money, functionality, etc.). Thus, she will only deviate from a given protocol if by doing so she can increase her utility.

While being self-enforcing is essential for a protocol to be adhered to, other desiderata can be conceived. In particular, a protocol might promote mutually beneficial collaboration between agents, in the sense that following it entails a collaboration between agents that improves their utilities with respect to the non-collaborative scenario. This self-enforcing collaboration is captured by the notion of *co-utility* [9, 8]. We say that a protocol is *co-utile* if helping the other agents that participate in the protocol to increase their utilities is also the way to increase one's own utility. That is, the protocol is built on the notion of mutual help as the best rational option.

We can find situations in which interactions are naturally self-enforcing and even co-utile. However, in many real-life cases, "negative" incentives (e.g., costs, lack of privacy, fear of strangers, etc.) may override positive incentives for the agents to follow the rules, let alone help each other. To tackle this problem, artificial positive incentives may need to be added to the agents' utility functions in order to compensate negative incentives. Specifically, in this paper we propose a general distributed mechanism that provides reputation as an artificial positive incentive (e.g. a stranger may be less feared if she has a good reputation). It turns out, though, that managing the reputation of the agents in a distributed way constitutes in itself a protocol that requires collaboration (e.g. to calculate, update and disseminate reputations); hence, *it is crucial that reputation management be designed to be co-utile*, so that the collaboration it implies be rationally sustainable. This is a significant aspect that differentiates our work from other reputation management solutions [12, 21], which rely on a central trusted authority [11], are application-oriented (e.g., file sharing, commercial transactions, social networks content generation) [27, 1, 28, 4] and/or just offer robustness against some tampering attacks [15, 12], but do not necessarily ensure the protocol to be rationally followed by the agents involved in the reputation calculation. In fact, as far as we know, no other work in the literature focuses on self-enforcing distributed reputation calculation and management.

In this paper, we first characterize the obstacles to designing self-enforcing and/or co-utile protocols that could be solved by appropriate reputation mechanisms; at the same time, we identify what a reputation mechanism should offer in order to qualify as appropriate. We then propose a general-purpose

distributed reputation model inspired in the well-known EigenTrust [15] mechanism. However, unlike EigenTrust, which is meant to filter inauthentic content in P2P file sharing scenarios, *our reputation protocol can be applied to a variety of scenarios and heterogeneous reputation needs, and it is itself co-utile, so that self-interested agents wish to follow it.* As a result of the latter, we create a "virtuous circle": our co-utile reputation management is the result of self-enforcing, mutually beneficial collaboration and provides reputations that are used as artificial positive incentives to turn protocols that were not originally self-enforcing (resp. co-utile) into self-enforcing (resp. co-utile) ones. Our theoretical contribution is illustrated with a detailed case study focused on co-utile P2P privacy-preserving query submission to a web search engine or a database.

The rest of the paper is organized as follows. Section 2 formalizes co-utility. Section 3 discusses the need for reputation mechanisms to achieve self-enforcing and/or co-utile protocols. In Section 4, we extend the EigenTrust reputation system [15] for reputations computed from non-binary opinions. Section 5 adapts the extended EigenTrust system to make it co-utile. Section 6 specifies how to incorporate reputations in existing protocols. Section 7 describes the aforementioned case study, which is evaluated formally and empirically. Section 8 lists conclusions and identifies lines for future research.

2. Co-utility: intuition and definitions

Co-utility aims at designing protocols whereby selfish agents engage in mutually beneficial collaboration. To this end, it is necessary that, by seeking an outcome that maximizes her utility, an agent helps the other agents improve their own utilities. When dealing with agents that are utility maximizers, as in co-utility, game theory is the natural framework for formalization. Game theory (see [19, 22]) models strategic interactions between a set of agents. A game is an abstraction of a scenario in which the elements relevant for strategic decision-making are explicit: the agents involved, the actions that each agent can take and the utility (payoff) that each combination of actions brings to each agent. The focus of this paper is on perfect-information games (a sequential game in which previously chosen actions are known). We use the following intuitive representation for them.

Definition 1 (Perfect-information game in extensive form)**.** A perfect-information game in extensive form is represented by a tree in which: (i) nodes are the points where decisions are made, (ii) each node is labeled with the name of the agent making the decision, (iii) edges going out from a node represent the available choices (actions) at that node, and (iv) each of the terminal nodes (leaves of the tree) is labeled with the tuple of payoffs that agents obtain when the node is reached.

Although we focus on perfect-information games, there are other types of games in which agents lack some information. For example, in Bayesian games the utility of an agent is known to her but the other agents are uncertain about

it. A way to model this uncertainty is to randomly assign a so-called type to each agent. The type of an agent completely determines her utility, and it is known to her but unknown (or at least uncertain) to the other agents. Even if in this section we define co-utility only for perfect-information games, it can also be defined for games where the information is not perfect. However, regardless of the class of games being considered, *ex post* checking whether co-utility has occurred (e.g. by an external auditor after the game is over) can be thought of being performed on a perfect-information game.

Let us now examine the connection between a protocol and a game. A protocol prescribes the interactions between agents for the completion of a given task; that is, among the actions available to agents in the underlying game, the protocol specifies the ones that are suitable for the completion of the task. Thus, we can view a protocol execution as a path that traverses the tree representing the game from the root node to a leaf (the sequence of actions prescribed by the protocol determines the edges in the path). In case the protocol allows for random choices between actions to be made at some nodes, we identify the protocol with a subtree rather than a path. In a node where a random choice can be made among multiple outgoing edges (actions), we label each outgoing edge with the probability of being chosen. The following definition summarizes the above ideas.

Definition 2 (Protocol)**.** Given a perfect-information game G in extensive form represented as a tree, a protocol is either a path from the root to a leaf or a subtree from the root to several leaves. In the latter case, alternative edges are labeled with probabilities of being chosen.

While a protocol prescribes a way in which agents should act in order to complete a task, agents may prefer to deviate from it. We are, therefore, interested in self-enforcing protocols, which are protocols from which agents have no rational incentive to deviate: in such protocols, any agent is better off by adhering to the protocol provided that the rest of agents also stick to it. That is, no agent can increase her utility by deviating from the protocol, provided that the other agents stick to it. Thus, a protocol is related to the game-theoretic notion of equilibrium in the way stated in the following definition.

Definition 3 (Self-enforcing protocol)**.** A protocol P on a game G is self-enforcing if no agent can increase her utility by deviating from P, provided that the other agents stick to P. Equivalently, at each successive node of the protocol path, sticking to the next action prescribed by the protocol (taking the next edge in the path) is an equilibrium of the remaining subgame of G (the subtree rooted at the current node). More technically, P on G is self-enforcing if and only if P is a subgame perfect equilibrium of G.

A self-enforcing protocol is said to be co-utile if it results in mutually beneficial collaboration between the participating agents. More specifically, a self-enforcing protocol P is co-utile when: i) the utility derived by each agent participating in P is strictly greater than the utility the agent would derive from not

participating, and ii) there is no alternative protocol P' giving greater utilities to all agents and a strictly greater utility to at least one agent. The first condition is needed to ensure that engaging in P is attractive for everyone (a self-enforcing protocol deters deviation, but it may not encourage participation if one or more agents do not gain anything from participating); the second condition can be rephrased in game-theoretic terms by saying that P is a Pareto-optimal solution of the underlying game. The following definition summarizes the above discussion.

Definition 4 (Co-utility)**.** A self-enforcing protocol P on a game G is co-utile if it is Pareto optimal and the utility derived by each participating agent is strictly greater than the utility the agent derives from not participating.

While co-utility seeks to promote collaboration between agents, its approach substantially differs from the one of cooperative game theory [5, 6]. In the latter, coalitions of agents are formed and the agents in a coalition coordinate their strategy to maximize the coalition payoff. This payoff is then divided among the agents. Assuming that the payoff can be divided is essential to motivate coalition formation. In contrast, in co-utility we assume that each agent acts autonomously and keeps to herself the payoff she obtains. This allows co-utility to deal with non-divisible payoffs, such as privacy and security. Moreover, in co-utility payoffs may be non-transferable and inexhaustible: e.g., an agent getting maximum privacy or security does not conflict with another agent getting it too.

On the other hand, because the goal of co-utility is the design of (co-utile) protocols, one may compare it with mechanism design. In mechanism design [13], the ultimate goal is to come up with mechanisms that lead to a previously defined socially desirable outcome. In this sense, co-utility is less demanding than mechanism design: it does not aim at enforcing a specific socially beneficial outcome but rather at promoting a mutually beneficial collaboration between agents. It may well happen that the mutually beneficial outcome of a co-utile protocol is also socially desirable (but this does not necessarily happen).

Another difference between mechanism design and co-utility is that the former requires preference alignment, whereas the latter does not. In mechanism design, the outcome is selected based on the preferences reported by the agents. Untruthful reporting of preferences by agents in an attempt to obtain a more desirable outcome is an important issue that must be addressed by mechanism design. This is usually tackled by requiring some payment from agents that is calculated to make misrepresentation of utilities unfruitful. This kind of mechanism to align the preferences of the individual agents with the socially desirable outcome is known as incentive-compatible mechanism [14]. In co-utility, rather than aligning preferences to the socially desirable outcome via incentives, we seek to promote the collaboration between agents that have complementary preferences. The situation in co-utility is less uncertain because decisions are made by the agents themselves rather than in a centralized way: when asked for collaboration, each agent decides which strategy is best for her.

A special case of co-utility is one in which the protocol does not only lead to a Pareto optimal payoff assignment to players, but it gives maximum payoff to all players. We call this strict co-utility:

Definition 5 (Strict co-utility)**.** A protocol P on a game G is strictly co-utile if the utility that each agent derives from participating in it is maximum.

It is easy to see that Definition 5 implies Definition 4.

Proposition 1. *If a protocol P on a game G is strictly co-utile then it is co-utile.*

Proof. We have to check that P is self-enforcing and Pareto optimal. Because each of the agents in P reaches her maximum payoff, no agent can increase her payoff by deviating from the protocol. Thus the protocol is self-enforcing. On the other hand, since each agent reaches her maximum payoff, the protocol is Pareto optimal; no other protocol can strictly increase the utility of any agent. □

In general, given any game G, there is no guarantee that the selfish behavior of the players will lead to co-utility, let alone strict co-utility. However, for some specific games, such a guarantee holds.

Proposition 2. *In a perfect-information game G where* all *the agents maximize their utilities in exactly the same set of terminal nodes (and only in these terminal nodes), selfish behavior by the agents will cause them to follow a strictly co-utile protocol.*

Proof. Let $\mathcal{M}$ be the set of terminal nodes where the utilities of all agents are maximized. Let l be the length of the shortest path from the root to a node in $\mathcal{M}$. We will prove that selfish behaviour leads to a node in $\mathcal{M}$ by induction over l.

If $l = 1$, there is a single action to be chosen and at least one node $m \in \mathcal{M}$ can be reached by the agent making the choice. This agent could certainly take a longer path (with more than one action to be chosen), but this path should equally lead to a node in $\mathcal{M}$ (otherwise the agent would obtain a suboptimal payoff).

We assume that the proposition is satisfied when $l \leq k$ and we need to show that it is also satisfied for $l = k + 1$. To apply the induction hypothesis we split the shortest path to $\mathcal{M}$ into two parts: P_1 (containing the k leading steps) and P_2 (containing the trailing step). The same reasoning used for $l = 1$ shows that the agent making the decision at $\mathit{leading}(P_2)$, the leading node of P_2, chooses to follow P_2. As we have determined the path that will be followed if $\mathit{leading}(P_2)$ is reached, we can simplify the game tree by removing all the subtrees rooted at $\mathit{leading}(P_2)$ and copying the utilities of m (which are the same as the utilities of any node in $\mathcal{M}$) to $\mathit{leading}(P_2)$. To show that m is reached in the original tree, it is enough to show that $\mathit{leading}(P_2)$ is reached in the simplified tree. But the latter is immediate by applying the induction hypothesis to P_1, because the terminal node of P_1 is $\mathit{leading}(P_2)$. □

If one wants to design a (strictly) co-utile protocol for a game G that admits none (as the sequential prisoners' dilemma), the only option is to modify the game G into G', for example by adding positive or negative incentives to it (rewards or penalties). This is what is done in the rest of this paper. Note that adding incentives can also be used less ambitiously, just to turn into self-enforcing a protocol that would otherwise not be so.

3. Towards co-utile protocols via reputation management

Co-utility or even strict co-utility can be reached in many practical situations. In the real world, for example, sharing one's car with one or several passengers can result in mutual benefits: passengers are able to reach their destination faster and the car driver needs to support less expense for the trip (because toll or gas fees are split among the passengers or because the driver can use facilities reserved for high occupancy vehicles or because of both reasons) [3]. In the information society, many peer-to-peer protocols follow the same pattern. File sharing protocols are based on the premise that, by sharing your own files and your upload bandwidth, you will also have access to a larger catalogue of files (shared by other users) with faster download speeds and greater availability than in centralized data storage systems [23]. Peer-to-peer protocols can also provide privacy protection to Internet users. For example, a user of a web search engine such as Google can hide her queries and search patterns from the search engine by requesting other peers to submit her queries (and return the results to her); this may be not only good for the requester's privacy, but also for the submitters' privacy, because the requester's query can be viewed as noise concealing the actual search interests of the submitters to the search engine [7].

In [9, 8] we showed how some of these protocols can be made co-utile. In all cases, this was possible because of the availability of agents with the appropriate types, that is, an appropriate combination of preferences/utilities (e.g., cost/time savings, privacy, functionality, etc.) that made their collaboration mutually beneficial. However, in practice, agent types can be very heterogeneous and "negative" utilities may dominate the benefits of agent collaboration, thereby precluding co-utility. For example, in the car-sharing scenario, agents may be reluctant to share a car if the passengers and/or the driver are complete strangers; in the query submission scenario, agents may be reluctant to submit queries from other users if that costs them bandwidth or money. Another possible obstacle to co-utility occurs when the self-enforcing behavior is *not* to collaborate rather than collaborate. For example, in an uncontrolled file sharing system, purely selfish agents would prefer to download files from others but not to share their own files (to avoid spending upload bandwidth), thus becoming "free riders"; eventually, if all agents become free riders, the file sharing system collapses.

Obstacles like the above ones should be tackled if one wants to design protocols that are co-utile for as many agent types as possible. A way to neutralize

the negative utilities that prevent co-utile collaboration is to incorporate artificial utilities in the form of rewards (that compensate the negative payoffs) or penalties (that discourage free riding). Note that rewards and penalties can also be used if, rather than obtaining a co-utile protocol, one merely wants to turn into self-enforcing a protocol that is not so.

Reputation is a very versatile artificial utility that can be used both as a reward and a penalty. In general, the reputation of an agent is the opinion of the community on the agent. On the one hand, reputation allows peers to build *trust*, which can neutralize the negative utilities related to mistrust (e.g., fear or reluctance in front of strangers). On the other hand, reputation also makes the agents accountable for their behavior: if an agent misbehaves, her reputation worsens and the other agents mistrust her more and more and are less and less interested to interact with her. In this manner, free riders and also malicious peers (who may try to subvert the system, even irrationally) may be identified and penalized (e.g., through limitation or denial of the service they try to abuse).

Many reputation mechanisms have been proposed in the literature for peer-to-peer communities, either centralized or distributed (see the surveys [12, 21]). However, since in our work reputation is used as a means to turn a protocol into co-utile or, at least, self-enforcing, the reputation calculation/management protocol itself should be distributed and strictly co-utile; otherwise, computing reputations would not be rationally sustainable and would not serve its purpose of inducing co-utility or self-enforcement in other decentralized P2P protocols.

Specifically, we want a reputation protocol with the following features, which should make it amenable to strict co-utility:

- *Decentralization.* Reputations should be computed and enforced by the peers themselves rather than a central authority. In this manner, we avoid depending on a trusted third party [11], which may be compromised and constitute a central point of failure (e.g., rational peers may be interested to attack this central authority).

- *Anonymity.* Reputation management should not rely on identifiers (e.g. IP addresses) that reveal the real identity of agents who help computing the reputation of other agents. Otherwise, the privacy loss may negatively affect the payoffs of collaborating in reputation management. Moreover, the possibility of creating coalitions between agents that know each other may facilitate collusion attacks to the reputation system. Surprisingly, most related works either support very limited anonymity or completely neglect it (see [24]).

- *Low overhead.* Reputation management should not imply a large expenditure of resources (e.g., bandwidth, storage, calculation); otherwise, these negative payoffs may dominate the benefits brought by reputation. To be more specific, a linear [15] or quasi-linear [25] calculation cost would be desirable.

- *Proper management of new agents.* Newcomers should not enjoy any reputation advantage; otherwise, malicious peers may be motivated to create new anonymous identifiers after abusing the system in order to regain the advantages of a good reputation.

- *Attack tolerance.* A number of attacks may be orchestrated in order to subvert the reputation system. Since we focus on rational selfish agents, we are interested in systems that are robust against "rational attacks". In particular, a rational selfish agent may try to subvert the protocol or manipulate the reputation calculation if, by doing so, she can increase her benefit; that is, if via subversion she can obtain a higher reputation at a lower cost than via "good behavior" in the underlying P2P scenario. Thus, the reputation system should implement measures to make the cost of such an attack unattractively high. According to the classification of attacks identified in [12], we want to avoid the following attacks:

  - *Self-promotion*, where agents are able to falsely increase their own reputations at a small or zero cost.
  - *Whitewashing*, in which agents circumvent the consequences of abusing the system to obtain an unfair benefit, for example by creating new "clean" identities or performing Sybil attacks. Analyses of whitewashing and Sybil attacks can be found in [10] and in [18], respectively.
  - *Slandering*, where agents may falsely lower the reputation of other agents if, by doing so, their own reputation becomes comparatively higher.
  - *Denial of service*, in which agents block the calculation and dissemination of reputation values. This may happen, for example, if the reputation calculation has a cost that agents deem higher than the benefits the calculation of their own reputation (by other agents) would bring to them.

After examining a number of decentralized reputation mechanisms [12], we selected, adapted and extended EigenTrust [15] in order to obtain our strictly co-utile reputation protocol. Eigentrust offers most of the desirable features identified above: distributed reputation calculation, low overhead, anonymity and robustness to attacks. This reputation scheme is designed to filter out inauthentic content in peer-to-peer file sharing networks. Its basic idea is to calculate a global reputation for each agent based on aggregating the local opinions of the peers that have interacted with the agent. If we represent the local opinions by a matrix whose component (i, j) contains the opinion of agent i on agent j, the distributed calculation mechanism computes global reputation values that approximate the left principal eigenvector of this matrix. Anonymity is achieved by means of a distributed hash table (DHT) that randomly links agents and reputation calculation duties. For unstructured networks, other solutions can be used, such as gossip-based reputation distribution protocols [29].

4. Reputation calculation model

The aim of the reputation mechanism is to compute global reputation values in a distributed way. These values are calculated by means of a protocol whereby the agents share their local reputation values. In this section, we extend EigenTrust to compute reputations based on non-binary opinions. The original EigenTrust system was designed for P2P file sharing, and the receiver's opinion on a file download is just binary: either the download has been satisfactory or not. We want to be able to compute reputations based on opinions that have several categories, or are even continuous. The continuous case may also accommodate a situation in which each agent quantifies the benefit/cost resulting from interacting with another agent.

4.1. Calculation of local reputations

The opinion of $\mathcal{P}_i$ on another agent $\mathcal{P}_j$ with whom $\mathcal{P}_i$ has directly interacted is the reputation s_{ij} of $\mathcal{P}_j$ local to $\mathcal{P}_i$. We define this value as the aggregation of payoffs (either positive or negative) that $\mathcal{P}_i$ has obtained from the set of transactions (Y_{ij}) performed with $\mathcal{P}_j$:

$$s_{ij} = \sum_{y_{ij} \in Y_{ij}} payoff_i(y_{ij}).$$

Payoffs may depend on the particular scenario, but the above calculation is general enough to accommodate them. Specifically, we can cope with the following scenarios that cover all the cases described in [12]:

- *Binary payoffs.* These are just binary values, either positive (+1) or negative (-1), according to whether the transaction with $\mathcal{P}_j$ has fulfilled or not $\mathcal{P}_i$'s goals. Binary payoffs make sense in P2P file sharing networks (i.e., the desired file has been obtained or not) or commercial transactions (e.g., in eBay, buyers rate transactions either as positive or negative).

- *Discrete or continuous opinions.* Payoffs may also measure the opinion on a transaction as a discrete or continuous magnitude. In this case, a certain transaction may generate reciprocal reputation updates by the involved agents (i.e., $\mathcal{P}_i$ on $\mathcal{P}_j$ and $\mathcal{P}_j$ on $\mathcal{P}_i$). Most transactions with a subjective outcome (other than the objective outcome of fulfilling or not the task/goal) should be measured in this way. For example, in a car-sharing network, passengers and drivers can rate each other to measure how pleasant the trip was or whether there were any unnecessary delays or overcosts; all of these dimensions can influence the utility of the involved peers and, thus, affect their willingness to collaborate in the future. Commercial transactions can also be rated in this way (e.g., Amazon provides discrete positive ratings), even though the rating is one-sided (not reciprocal). In some cases, the opinion can be the aggregation of several transaction components (e.g., reviews, comments, etc.) [16].

- *Costs.* In the previous cases, reputation values are potentially unbounded (i.e., there is not a limited reputation budget to be distributed among the peers of the network) and independent between the agents. Alternatively, payoffs could reciprocally measure the cost incurred/caused by the agents when helping others/being helped in fulfilling a task/goal. For example, in a P2P file sharing network, if $\mathcal{P}_i$ provides a file to $\mathcal{P}_j$, the payoff of the former with respect to the latter would be a negative value measuring the bandwidth spent by $\mathcal{P}_i$ when uploading the file; likewise, the payoff of $\mathcal{P}_j$ (who has received the desired file) with respect to $\mathcal{P}_i$ would be the inverse positive value. In the privacy-preserving query submission scenario discussed in Section 3 a similar criterion can be applied: if performing a query to the database has a cost (e.g. money) for the submitter, this should be reflected as a positive reputation payoff for the submitter and as a negative reputation payoff for the query originator. In these scenarios, reputation values can measure the difference between the cost incurred by an agent $\mathcal{P}_i$ when helping others and the cost incurred by others when helping $\mathcal{P}_i$. A reputation greater than 0 shows that the agent has helped others more than what others have helped her. Unlike in the previous cases in which reputation values were unbounded, here reputations are positively/negatively balanced across the members of the network; thus, a reputation value around 0 indicates a fair allocation of costs among peers.

It is important for local reputation values computed by the different peers to measure the payoff of similar transaction outcomes in a similar way. For objective outcomes (such as the fulfillment of a task or the incurred cost), this is not problematic; however, for subjective opinions the designer of the reputation system may provide some rules to control the ratings. For example, eBay recently implemented a "subjective" rating[2] which buyers can use to rate sellers from 5 stars down to 1 star regarding several aspects of the transaction (e.g., delivery speed of the goods, shipping charges, etc.); however, eBay associates each value to an objective criterion (e.g., a 5-star shipping charge means free shipping, whereas less than 3-star means a charge higher than the real shipping cost).

4.2. Computing global reputations

We review here how EigenTrust computes global reputations from local reputations; recalling this is needed for the reader to understand what we propose in Sections 5 and 6. In order to properly aggregate the local reputation values computed by each peer, a normalized version c_{ij} is first calculated as follows:

$$c_{ij} = \frac{\max(s_{ij}, 0)}{\sum_j \max(s_{ij}, 0)}.$$

[2] `http://pages.ebay.com/help/feedback/detailed-seller-ratings.html#calculated`

In this manner, the normalized local reputation values lie between 0 and 1 and their sum is 1. In other words, each agent has a reputation budget of only 1 that she has to split among her peers proportionally to her positive experiences (negative experiences are truncated to 0). This makes all agents equal contributors to the global reputation and avoids dominance by agents with a larger number of experiences. Moreover, this normalized calculation deters peers from colluding by assigning arbitrarily high values to each other. Finally, the fact that negative reputation values are truncated to 0 prevents selfish agents from assigning arbitrarily low values to good peers. We will later discuss how the protocol can also withstand tampering with reputation values.

A side effect of the truncation of negative values is that reputation values do not distinguish between agents with whom $\mathcal{P}_i$ had a bad experience (negative local reputation) from those with whom $\mathcal{P}_i$ has not interacted so far. Even though this may be seen as a drawback, it also can be viewed as a strength: newcomers (i.e., agents who have not yet interacted with anyone) do not have any reputation advantage, because their reputation is indistinguishable from the one of misbehaving agents. As a result, a selfish agent has no incentive to take a new virtual identity in order to "clean" her reputation after abusing the system (e.g., refusing to help others, in order to minimize her own costs). Likewise, newcomers will become instantly motivated to positively contribute to the system in order to earn the minimum reputation that other agents would require from them.

We can thus see that normalizing local reputation values biases the system towards positive reputation; that is, agents need a minimum *positive* reputation value in order to be trusted by peers, which requires them to help others/contribute to the system first on order to be able to get help later.

In this setting, local reputation values are disseminated and aggregated through the network peers following a transitive reputation algorithm, in order to obtain the global reputation value of each agent. This is the main idea of EigenTrust. Any agent $\mathcal{P}_i$ can compute $\hat{t}_{ik}^{(0)}$, an approximation of the reputation of a potentially unknown peer $\mathcal{P}_k$, by asking the peers with whom $\mathcal{P}_i$ has interacted ($\mathcal{P}_j$) for their local reputation w.r.t. $\mathcal{P}_k$, that is c_{jk}. Since $\mathcal{P}_i$ has already computed the local normalized reputation w.r.t. $\mathcal{P}_j$, that is c_{ij}, $\mathcal{P}_i$ can compute a local estimate of the reputation t_{ik} of $\mathcal{P}_k$ by using c_{ij} to weight $\mathcal{P}_j$'s local reputation; specifically, $\hat{t}_{ik}^{(0)} = \sum_j c_{ij}c_{jk}$. Thanks to the local normalization, $\hat{t}_{ik}^{(0)}$ takes values between 0 and 1. Observe that if we call $\mathbf{c}_i = (c_{i1}, \ldots, c_{in})^T$ and $\mathbf{C} = [c_{ij}]$, then $\hat{\mathbf{t}}_i^{(0)} = \mathbf{C}^T\mathbf{c}_i$, where $\hat{\mathbf{t}}_i^{(0)} = (\hat{t}_{i1}^{(0)}, \ldots, \hat{t}_{in}^{(0)})^T$. If every agent $\mathcal{P}_i$ computes $\hat{\mathbf{t}}_i^{(0)}$, in the next iteration $\mathcal{P}_i$ can compute $\hat{\mathbf{t}}_i^{(1)} = \mathbf{C}^T(\mathbf{C}^T\mathbf{c}_i)$. After m iterations, $\mathcal{P}_i$ will compute $\hat{\mathbf{t}}_i^{(m-1)} = (\mathbf{C}^T)^m\mathbf{c}_i$. Under the assumptions that $\mathbf{C}$ is irreducible and aperiodic [15], in an ideal and static setting the succession of reputation vectors computed by any peer will converge to the same vector for every peer, which we call $\mathbf{t} = (t_1, \ldots, t_n)^T$ and is the left principal eigenvector of $\mathbf{C}$. The j-th component of $\mathbf{t}$ represents the global reputation of the system on each agent $\mathcal{P}_j$.

Computing the global reputation values by the above method is not efficient because the communication complexity is very high. In the following section we present a protocol to compute global reputation values that emulates the above method in a distributed and more efficient way.

5. Co-utile distributed reputation calculation protocol

We adapt and extend the EigenTrust secure protocol [15] in order to obtain a co-utile distributed protocol for computing global reputation values. The core idea of the EigenTrust secure protocol is that the reputation value of an agent $\mathcal{P}_i$ is computed by other agents in order to prevent manipulation and minimize the action of malicious peers. This computation is based only on the local reputation reported by those agents with whom the agent $\mathcal{P}_i$ interacted directly.

In our proposal, the computation of t_i (the global reputation of agent P_i) is based on the experience of the agents in A_i, which is the set of agents that provided help to or received help from $\mathcal{P}_i$. Hence, the global reputation values are computed in a distributed way according to the experiences of the agents in the network.

We assume that each agent $\mathcal{P}_i$ has an initial reputation value $t_i^{(0)}$ that is based on previous experiences or is given by default. Each agent has a number M of score managers that will compute her reputation value. These score managers are defined according to a distributed hash table (DHT), which maps each agent to a set of several agents determined by hash functions $h_0, h_1, \ldots, h_{M-1}$. In this way, $h_0(ID_i), \ldots, h_{M-1}(ID_i)$ are the pseudonyms within the P2P network of the agents computing the reputation value of $\mathcal{P}_i$, where ID_i is $\mathcal{P}_i$'s identifier. Note that the use of pseudonyms provides anonymity, and the use of a DHT prevents anyone (in particular $\mathcal{P}_i$) from choosing a particular pseudonym as a score manager for $\mathcal{P}_i$.

With the above hash mapping, on average every agent is the score manager of M agents, so the work of the agents in the reputation mechanism is balanced. Let D_i be the set of agents for whom $\mathcal{P}_i$ is a score manager; we will also call D_i the set of daughters of $\mathcal{P}_i$. During the computation of the reputation values, the score manager of $\mathcal{P}_d$, say $\mathcal{P}_i$, learns A_d, that is, the set of agents that provided help to or received help from $\mathcal{P}_d$. Then $\mathcal{P}_i$ receives the trust assessments of $\mathcal{P}_d \in \mathcal{D}_i$ sent by the agents in A_d. The terms c_{ji} for $j \notin A_i$ are zero. Then, $\mathcal{P}_i$ engages in an iterative refinement to compute the reputation of every $\mathcal{P}_d \in D_i$, based on the score c_{jd} on $\mathcal{P}_d$ by each $\mathcal{P}_j \in A_d$ weighted by the score managers of $\mathcal{P}_j$ (the weight is the current reputation $t_j^{(k)}$ of $\mathcal{P}_j$ held by the score managers of $\mathcal{P}_j$). As a result, no agent $\mathcal{P}_j$ can directly influence the reputation of any other $\mathcal{P}_d$; everything is mediated by the score managers of $\mathcal{P}_j$, who together act as a sort of distributed trusted third party.

The above computations are described in Protocol 1. The main differences between this protocol and the one in [15] are: i) we rely for most of the calculation on the score managers (Step 7), thereby increasing the redundancy of the

computation of the reputation values and protecting this computation against malicious agents; and ii) our definition of the agents that provide the local reputation values is more general, since we include any agent that has interacted with the daughter agent (Steps 3, 5, 9, 10 and 11).

Protocol 1 CO-UTILE COMPUTATION OF REPUTATION VALUES

1: **for all** agent $\mathcal{P}_i$ **do**
2: Submit local reputation values $\mathbf{c}_i$ to all score managers at positions $h_m(ID_i)$, for $m = 0, \ldots, M-1$;
3: Collect local reputation values $\mathbf{c}_d$ and the set A_d for all daughter agents $\mathcal{P}_d \in D_i$;
4: **for all** $\mathcal{P}_d \in D_i$ **do**
5: Query all the score managers of $\mathcal{P}_j \in A_d$ for $c_{jd}t_j^{(0)}$
6: $k := -1$
7: **repeat**
8: $k := k + 1$
9: Compute $t_d^{(k+1)} = c_{1d}t_1^{(k)} + c_{2d}t_2^{(k)} + \ldots + c_{nd}t_n^{(k)}$;
10: Send $c_{dj}t_d^{(k+1)}$ to all the score managers of $\mathcal{P}_j \in A_d$;
11: Wait for all score managers of $\mathcal{P}_j \in A_d$ to return $c_{jd}t_j^{(k+1)}$;
12: **until** $|t_d^{(k+1)} - t_d^{(k)}| < \epsilon$ // Parameter $\epsilon > 0$ is a small value
13: **end for**
14: **end for**

Once a reputation manager computes a reputation on a daughter agent using Protocol 1, she keeps the reputation value for that agent until the next protocol execution. Protocol 1 is meant to be run periodically, in order for reputations to stay up-to-date. The reputation update period can be set depending on the activity of the agents, in order to obtain faster updates when the frequency of agent interactions increases. Ideally, the protocol should be run in parallel and asynchronously with respect to the agent interactions.

After the computation of the global reputation values, if an agent $\mathcal{P}_i$ needs the reputation value t_j of $\mathcal{P}_j$ in order to decide whether to collaborate with him, she can query the M score managers of $\mathcal{P}_j$ for his reputation. The M values obtained from the M score managers should be the same, because the inputs of the score managers are the same. However, if some values are different (e.g., if some score managers or agents have altered the computation for some reason), $\mathcal{P}_i$ can take as t_j the most common value among the ones sent by the score managers. If a score manager $\mathcal{P}_k$ does not answer to a query by $\mathcal{P}_i$, then $\mathcal{P}_i$ will assume that $\mathcal{P}_k$ is not active and she will set $c_{ik} = 0$.

In addition to the global reputation, the experience of $\mathcal{P}_i$ with respect to $\mathcal{P}_j$ is also reflected in the local value c_{ij}, if available (i.e., if $\mathcal{P}_i$ and $\mathcal{P}_j$ have already interacted). In some cases, local and global reputation values may not be coherent because the latter is the aggregated version of the former. In general, it is better for $\mathcal{P}_i$ to consider both local reputations $\mathbf{c}_i$ and global reputations $\mathbf{t}$ to make decisions about collaboration; a conservative criterion would be to use

the lowest among the local and global reputations.

5.1. Co-utility analysis

Within the co-utility framework, we consider that agents are rationally selfish. For the calculation of reputations, this means that: i) agents want their reputation to be computed (this is their main utility); ii) they are interested in maximizing their reputation (so that it is higher than the reputations of other peers) by any means (i.e., either by correctly following the protocol or by deviating from it). Ideally, the reputation calculation protocol should be self-enforcing (hence discouraging deviation); what is more, it should be co-utile, given that all agents obtain their correct reputation as the outcome utility of the protocol execution. In the sequel, we analyze Protocol 1 and, for each step, we describe the options available to the agents and we justify why correctly following the protocol is the rational choice and, thus, why the protocol is self-enforcing and, in particular, co-utile.

Generally speaking, Protocol 1 encourages the agents to collaborate because the impact of their opinions on the computation of the global reputation values increases when they are active (that is, present in as many sets A_* as possible). On the other hand, the protocol is robust against *self-promotion attacks*, because the agents that actually compute the global reputation values of an agent $\mathcal{P}_i$ (her score managers) are chosen randomly, and they use the information provided by the score managers of the agents with whom $\mathcal{P}_i$ has interacted; thus, the agents cannot manipulate their own reputation values because they do not have direct control on their calculation. Also, Protocol 1 is robust against *whitewashing attacks*: since the local reputation values are truncated to 0, the system does not distinguish between agents with whom there was a bad experience (whose negative s_{ij} is truncated to 0) and those with whom no one has interacted so far. In this way, selfish agents have no incentive to create new virtual identities to reset their bad reputation because, in practice, this does not make them better off.

Let us now be more specific and go step by step. At Step 2, agent $\mathcal{P}_i$ acts as a reputation assessor and sends her local reputation values $\mathbf{c}_i$ to the reputation managers. Any peer $\mathcal{P}_i$ has a rational interest in forwarding to the rest of peers the fair local reputations c_{ij} she has awarded to peers $\mathcal{P}_j \in A_i$. We next justify why:

- If $\mathcal{P}_i$ fails to forward c_{ij}, then c_{ij} is taken as zero, and hence the reputation t_j of $\mathcal{P}_j$ is lower than due. Now, since $\mathcal{P}_j \in A_i$ implies $\mathcal{P}_i \in A_j$, by the expression in Step 9, a lower t_j also results in a lower reputation t_i of $\mathcal{P}_i$, *ceteris paribus*.

- If $\mathcal{P}_i$ forwards a reputation c'_{ij} less than the real c_{ij}, the argument is the same as in the previous item. Hence, Protocol 1 is robust against *slandering attacks by assessors*.

- If $\mathcal{P}_i$ forwards a reputation c'_{ij} greater than c_{ij}, this increases t_j and t_k for agents $\mathcal{P}_k \in A_j$. Although $\mathcal{P}_i$ is also in A_j and thus benefits from the

increase, *ceteris paribus* t_i will increase less than t_j and similarly as t_k for $k \neq i, j$. Hence, $\mathcal{P}_i$ will be in a worse relative position with respect to $\mathcal{P}_j$ and possibly other peers.

From Step 3 onwards, agent $\mathcal{P}_i$ acts as a reputation manager and collects the local reputations awarded by all his daughter agents $\mathcal{P}_d$, which in turn allows $\mathcal{P}_i$ to learn the set A_d of peers with whom $\mathcal{P}_d$ has interacted. The same justification above that $\mathcal{P}_i$ is rationally interested to report fair local reputations ensures that $\mathcal{P}_d$ will report all her local reputations fairly. On the other hand, if the score manager $\mathcal{P}_i$ does not collaborate to compute the global reputation of his daughters (*denial-of-service attack*), the requester of t_d will receive no answer from $\mathcal{P}_i$; in this case, the requester will consider that $\mathcal{P}_i$ not active, and will remove $\mathcal{P}_i$ from the list of available agents. Hence, $\mathcal{P}_i$'s own reputation value will be decreased in the following iteration of the reputation calculation protocol. Thus, a rational $\mathcal{P}_i$ is not interested in denying service.

The iteration starting in Step 7 involves the score manager $\mathcal{P}_i$ and the score managers of peers in A_d. For the same reasons given above to justify that $\mathcal{P}_i$ is not interested in denying service, the score managers of peers in A_d are not interested either.

Finally, even if not denying service, a score manager might send a wrong value when queried by an agent (Steps 5, 10 and 11). Assuming the security parameter M has been chosen so that the number of malicious agents can be safely considered to be less than $M/2$, such misbehavior will have no effect because the final value taken by the requesting agent is the majority value reported the score managers. Two remarks are in order here:

- The score managers are chosen by means of a hash function, that is, randomly. This makes it unlikely for them to collaborate in reporting a common manipulated t_d. If all wrong values can be assumed to be different, malicious score managers could be as numerous as $M - 2$ and there would still be two identical correct values of t_d, which would make malicious behavior ineffectual.

- If the requester can tell which reported reputations are wrong, he can lower the local reputation she awards to the corresponding score managers.

Hence the distributed nature of the algorithm and the redundancy of the computation of the reputation values protect agents against *slandering attacks by the score managers*.

From the discussion above, we see that Protocol 1 is self-enforcing. Also, all agents are guaranteed to get as high a reputation as they fairly deserve, so the protocol is strictly co-utile.

6. Reputation-based co-utility enforcement

As discussed in Section 3, negative payoffs resulting from helping others (e.g. the bandwidth spent on transferring a file to another peer) may dissuade rational

selfish agents from collaborating, thereby preventing a collaborative protocol from being co-utile. Negative payoffs associated to the execution of the s-th protocol step by an agent $\mathcal{P}_i$ can be formally added to the agent's aggregated utility $u_{i,s}$ as a negative cost $o_{i,s}$ weighted by a coefficient α_i^o expressing the importance attached by $\mathcal{P}_i$ to the cost:

$$u'_{i,s} = u_{i,s} - \alpha_i^o \cdot o_{i,s}$$

If, as a result of the weighted negative cost, the aggregated utility $u'_{i,s}$ is negative, the agent will rationally choose not to execute the s-th protocol step.

Reputation can be used as a positive incentive that neutralizes negative payoffs, as long as it is aligned with the nature of the negative payoffs. For example, if costs represent the reluctance of agents to collaborate with each other (because of the fear to interact with strangers or misbehaving agents), reputations can reflect how much a certain agent is trusted based on the outcome of past interactions with her. In this case, positive reputations t_j of other peers $\mathcal{P}_j$ can be used to neutralize the negative costs incurred by a certain agent $\mathcal{P}_i$ when executing a protocol step that would be beneficial for $\mathcal{P}_j$. Thus, $\mathcal{P}_j$'s reputation at the time of running step s, say $t_{j,s}$, is subtracted from the cost, as follows:

$$u'_{i,s} = u_{i,s} - \alpha_i^o \cdot (o_{i,s} - \alpha_i^t \cdot t_{j,s}), \tag{1}$$

where α_i^t is explained next. Putting aside the other atomic utilities, the cost $o_{i,s}$ incurred by agent $\mathcal{P}_i$ can be seen as a threshold that specifies the *minimum reputation* value that $\mathcal{P}_j$ should have in order for $\mathcal{P}_i$ to follow the protocol that will also help $\mathcal{P}_j$. In this context, α_i^t is a coefficient that allows coherently aggregating cost and reputation values (because global reputations are normalized in the range [0..1], whereas the cost can be an unbounded value). In this way, the α_i^o coefficient now weights the difference between the actual cost $o_{i,s}$ and the agent reputation $t_{j,s}$ that compensates the former.

If several different costs (and reputations) can be associated to an action (e.g., in eBay, sellers are rated independently according to delivery speed, shipping costs, communication, etc.), this approach can be generalized in at least two different ways:

- *Aggregation.* For each peer, costs are aggregated into a single cost, and reputations are aggregated into a single reputation, so that the resulting aggregated reputation can compensate the resulting aggregated cost as per Expression (1).

- *Separation.* For each peer, each cost and each reputation are separately managed. In this case, the utility functions for the agents considering to collaborate will incorporate one compensation term for each cost type and corresponding compensating reputation type. That is, instead of subtracting a single term as in Expression (1), one would subtract one term for each cost-reputation pair being considered.

Reputations can also be viewed as atomic utilities that agents wish to increase (because by doing so they also increase the willingness of other agents to collaborate with them). As a result, an agent $\mathcal{P}_i$ may consider as an additional atomic utility her own reputation gain resulting from executing a certain protocol step s:

$$u'_{i,s} = u_{i,s} + \alpha_i^t \cdot (\hat{t}_{i,s+1} - t_{i,s}). \tag{2}$$

In Expression (2), $t_{i,s}$ is the reputation of $\mathcal{P}_i$ before running step s and $\hat{t}_{i,s+1}$ is an estimate of $\mathcal{P}_i$'s reputation $t_{i,s+1}$ after running step s. If agents rate the outcomes of actions in an objective and/or similar way (e.g. spent bandwidth, binary fulfillment of an action, etc.), then the exact value of $t_{i,s+1}$ can be anticipated; otherwise, if reputation values result from subjective opinions (e.g., how pleasant was a trip in a shared car), only the estimate $\hat{t}_{i,s+1}$ can be computed. This shows the importance of agents measuring reputations in a homogeneous way.

With the utility of Expression (2), agents are motivated to increase their reputations indefinitely. However, if reputation values measure the difference between the cost incurred to help other agents and the cost caused to other agents for the help received from them (as in the file sharing scenario), agents are no longer interested in systematically increasing their reputation (because of the incurred costs). They will rather aim at maintaining a reputation value that is barely sufficient to obtain collaboration from other peers (e.g. slightly greater than 0 in the file sharing scenario, which means that the agent has helped others more than what others have helped him). Let such target reputation value be t_{target}. Then the utility expression that represents the interest of the agent in achieving this value (but not more than it) is:

$$u'_{i,s} = u_{i,s} + \alpha_i^t \cdot \min((\hat{t}_{i,s+1} - t_{i,s}), (t_{target} - t_{i,s})). \tag{3}$$

In plain words, Expression (3) reflects that an agent that accumulates more reputation than the target does not obtain any more utility than if she sticks to the target. Hence, if step s of the protocol takes the agent beyond the reputation target, she will not be interested any more in continuing to help others in the next step, because helping is costly and she does not need to increase her reputation further. If the agent subsequently requests and obtains help from other peers, her reputation will decrease; as soon as it falls below the target, the agent will be again interested to help others in order to reach the target once more. Thus, agents with utility given by Expression (3) tend to stick to the target reputation, rather than systematically increasing their reputation. If reputation is the difference between costs borne and costs caused to others, everyone's target reputation will be just slightly above 0, which results in a fair distribution of costs.

The additional atomic utilities considered in the previous expressions can also be combined. For example, an agent may evaluate both the reputation of the peer she is helping (with regard to the cost this help will cause) and also the reputation increase that this action is likely to bring to her own reputation.

Depending on the importance attached by each agent to each atomic utility and the values of these utilities/reputations, agents with low reputations may become motivated to collaborate more (in order to increase their reputations), even with agents with not so high reputations. This feature is useful in the file sharing scenario, for example, in which reputations measure the difference between the spent upload and download bandwidths: agents with a low reputation (i.e., those who have downloaded more data than they have uploaded) will be more motivated to fulfill file requests by other peers; otherwise, agents with good reputation are likely to turn down their file requests (because of the low reputation of the requesters).

The reputation calculation protocol enables all agents in the network to faithfully demonstrate their reputation/collaboration record in an anonymous way. Thus, the global reputation t_i achieved by each agent (which can influence the decision of other peers to collaborate) is not a component of the secret type of an agent (which could be manipulated to cheat other peers into changing their strategies), but rather it is public global information that is both accessible and reliably verifiable by the other peers at all times. This fact, in turn, ensures that the reputation atomic utility is a reliable incentive to turn a non-co-utile protocol into a co-utile one.

7. Case study

In this section we discuss and illustrate the application of a reputation mechanism to enforce co-utility in a specific peer-to-peer protocol.

7.1. Co-utile privacy-preserving query submission protocol

We focus our case study on the privacy-preserving query submission protocol introduced in Section 3. Nonetheless, the analysis and the conclusions we will give can be applied to other similar protocols, such as P2P file sharing.

Consider an agent who wants to submit queries to a web-search engine (WSE) or a database without the latter learning her interests. That is, the agent wants to avoid getting profiled by the WSE. The agent's defense consists in making her interest profile flat (diverse) enough so that the WSE cannot determine her interests. In essence, to mask her profile this agent must hide the real queries she is interested in in a set of (real or fake) queries about diverse topics she is not interested in.

If there are several agents with similar privacy interests, an agent can mask her profile by asking another agent to submit her query to the WSE rather than submitting it herself. Also, the agent can submit to the WSE real queries originated by other agents. In this way, a rational cooperation between peer agents emerges [7] and all of them manage to blur their profile of interests without having to submit fake queries (which are easier to filter out by the WSE).

The proposed protocol works as follows: instead of each agent always submitting her queries directly, which may decrease her privacy, a query initiator

may choose to forward her query to another randomly chosen agent. The receiving agent, in turn, may decide to submit the query to the WSE (and return the result to the query initiator) or to refuse it. The receiver will be interested in submitting the query if the query topic helps him to flatten his profile towards the WSE and, thus, improves his privacy. If the receiver refuses the query, the query initiator can ask another randomly chosen agent until someone interested in submitting the query is found. Only if all available peers refuse her request, the initiator will submit her query herself.

The utility of the initiating agent $\mathcal{P}_i$ (i.e., the one who wants to have a query q answered), can be defined as the aggregation of the functionality and privacy atomic utilities, which are a function of the query q:

$$u_i(q, f_i(\cdot), Y_i, \alpha_i^H) = f_i(q) + \alpha_i^H \cdot \Delta(H(Y_i), q). \tag{4}$$

The first term, $f_i(q)$, represents the functionality that the agent gains from having q answered, which is 1 if the query is answered and 0 otherwise. The second term quantifies the privacy gain or loss w.r.t. the WSE as a result of the submission of the query to the WSE (by $\mathcal{P}_i$ herself or by another agent). Specifically, Y_i is $\mathcal{P}_i$'s profile of submitted queries (which is what the WSE sees on $\mathcal{P}_i$); it is represented as a distribution of topics extracted from the queries submitted so far by agent $\mathcal{P}_i$. From $\mathcal{P}_i$'s perspective, Y_i characterizes the exposure level (i.e., privacy loss) of the agent's interests towards the WSE. The function H is Shannon's entropy and, thus, $H(Y_i)$ measures the uncertainty (i.e., privacy) of the profile. Note that the highest privacy level is reached when the profile is empty, because no information about the profile has been disclosed. For non-empty profiles, the maximum entropy/privacy is reached if all topics appear with the same frequency, because the agent's interests remain unspecific. The $\Delta(H(Y_i), q)$ function represents the variation (positive or negative) of the agent's privacy in terms of the profile's entropy $H(Y_i)$ as a result of $\mathcal{P}_i$ submitting herself the query q to the WSE; if the query is finally submitted by any other agent, $\Delta(H(Y_i), q) = 0$, because the WSE will not learn anything new about the initiator.

Finally, the α_i^H coefficient normalizes the scales of functionality and privacy (so that these heterogeneous features can be coherently compared) and also weights their relative importance to agent $\mathcal{P}_i$. By properly configuring this coefficient, we can make functionality dominate (which is a natural choice).

The actions available to the initiator $\mathcal{P}_i$ are either to submit q herself to the WSE or to forward the query to another agent $\mathcal{P}_j$ (the responder); that is, $A_i = \{submit, forward\}$.

For the responder $\mathcal{P}_j$, the only utility that matters is his own privacy w.r.t. the WSE and how it is modified by submitting the query q of the initiator $\mathcal{P}_i$. Thus, to derive the utility for $\mathcal{P}_j$, we set $f_j(q) = 0$ in Expression (4) and we obtain the following simplified expression:

$$u_j(q, Y_j) = \Delta(H(Y_j), q). \tag{5}$$

The actions available to agent $\mathcal{P}_j$ are to submit the query of $\mathcal{P}_i$ to the WSE or to refuse submitting it, that is, $A_j = \{submit, refuse\}$.

The privacy-preserving query submission protocol is executed when the initiator decides to *forward* her query to a random responder and the latter decides to *submit* it to the WSE and return the response. The above happens if and only if the initiator considers that submitting her query herself decreases her privacy towards the WSE (i.e., $\Delta(H(Y_i), q) < 0$) and the responder considers that submitting the initiator's query improves his privacy towards the WSE (i.e., $\Delta(H(Y_j), q) > 0$). In other words, the protocol is executed if and only the initiator chooses her maximum-payoff action in terms of functionality and privacy (that is, to forward her query to the responder) and the responder chooses her maximum-payoff action in terms or privacy (that is, to submit the query). Thus, the protocol is executed if and only if it is strictly co-utile (according to Definition 5).

7.2. Negative payoffs and reputations

Co-utility in the scenario depicted above may be hampered by additional hindrances that may arise in a real setting. In particular, submitting a query q may result in a negative cost (e.g., spent bandwidth or money) to the agents. In this case, the utility functions for the initiator and responder agents need to incorporate this cost $o(q)$ as an additional negative atomic utility:

$$u_i(q, f_i(\cdot), o_i(\cdot), Y_i, \alpha_i^H, \alpha_i^o) = f_i(q) + \alpha_i^H \cdot \Delta(H(Y_i), q) - \alpha_i^o \cdot o_i(q). \quad (6)$$

$$u_j(q, o_j(\cdot), Y_j, \alpha_j^o) = \Delta(H(Y_j), q) - \alpha_j^o \cdot o_j(q). \quad (7)$$

For the initiator agent, the cost $o_i(q)$ constitutes an additional incentive to prefer forwarding the query (which results in $o_i(q) = 0$) rather than submitting it herself. For the responder agent, if the privacy gain $\Delta(H(Y_j), q)$ brought to him by the query q does not dominate (considering the weighting coefficient) the cost $o_j(q)$ of the query, the outcome of the utility function is negative and submitting the query "for free" is not rational for him anymore. Thus, in this case, the protocol is no longer self-enforcing, let alone co-utile.

In order to neutralize this negative payoff, agent reputations can be used to ensure collaboration in a way that the costs associated to query submission are fairly distributed throughout the network.

Specifically, we propose to compute the reputation t_i of an agent $\mathcal{P}_i$ as the difference between the cost incurred by $\mathcal{P}_i$ when submitting queries on behalf of other agents and the cost incurred by other agents as a result of submitting $\mathcal{P}_i$'s queries. Assuming the cost of a query submission is uniform for all agents (which implies that agents calculate and evaluate reputations in a similar way), if an agent $\mathcal{P}_i$ has a reputation t_i greater than 0, it means that she has helped others more than what others have helped her.

To evaluate whether the collaboration record of an initiator agent $\mathcal{P}_i$ compensates the cost of helping her, a responder agent $\mathcal{P}_j$ can take into account

the reputation t_i of the initiator agent in the way described by Expression (1) above. In our case study, this yields:

$$u_j(q, o_j(\cdot), Y_j, t_i, \alpha_j^o, \alpha_j^t) = \Delta(H(Y_j), q) - \alpha_j^o \cdot (o_j(q) - \alpha_j^t \cdot t_i). \tag{8}$$

According to this expression and leaving aside the privacy gain, $\mathcal{P}_j$ will get a positive utility outcome if the reputation t_i of the initiator compensates the negative cost of the query $o_j(q)$. Likewise, $\mathcal{P}_j$ will decide to refuse submitting queries from any agent $\mathcal{P}_i$ whose reputation (multiplied by the corresponding weight) is less than the relative cost of the query $o_j(q)$. Since the cost is an unbounded value and the reputation is a normalized value, the α_j^t coefficient is necessary to make both values comparable.

Note, however, that the relative importance given to privacy with respect to the cost/reputation difference (weighted by the α_j^o coefficient) may even compensate a negative difference, because an agent may still help a peer with low reputation if the former attaches a lot of importance to her own privacy.

Reputations in our work are biased towards positive values; that is, in general, initiator agents only get help from other peers if they have already helped other agents (i.e., submitted their queries) more than what others have helped them. In this way, non-collaborative agents become motivated to help others in order to increase their own reputation beyond the relative cost of their queries; otherwise they will be forced to submit their queries themselves, thereby incurring two negative payoffs: query submission cost and loss of privacy. Since the sum of these negative payoffs will severely decrease the agent's utility, helping others becomes the rational choice (helping others certainly has a cost, but it also increases both the reputation and the privacy of the helper).

Thanks to the compensation of costs via reputation, the above reputation-based query submission protocol is strictly co-utile in spite of the query costs. To see this, note that, if the responder $\mathcal{P}_j$ agrees to submit, it is because this is the action among submitting or refusing the initiator's query that brings the maximum utility to $\mathcal{P}_j$. In this case, the outcome for the initiator $\mathcal{P}_i$ is also maximum (because she avoids the privacy loss and the cost of submitting her query).

Ideally, to balance the costs and the social welfare of all agents, these should achieve and maintain a bounded reputation t_{target} slightly greater than 0 that is sufficient to get help from other peers. In this respect, as discussed in Section 6, agents can view their own reputations as atomic utilities: they should measure the reputation gain that each action of the protocol brings them and how far their actual reputation is from the target reputation t_{target} they want to achieve. Thus, by adapting and adding Expression (3), the aggregated utility functions are as follows:

$$\begin{aligned} &u_i(q, f_i(\cdot), o_i(\cdot), Y_i, \alpha_i^H, \alpha_i^o, \alpha_i^{t'}) = \\ &= f_i(q) + \alpha_i^H \cdot \Delta(H(Y_i), q) - \alpha_i^o \cdot o_i(q) + \alpha_i^{t'} \min((t_{i,1} - t_{i,0}), (t_{target} - t_{i,0})). \end{aligned} \tag{9}$$

$$u_j(q, o_j(\cdot), Y_j, \alpha_j^o, \alpha_j^t, \alpha_j^{t'}) =$$
$$= \Delta(H(Y_j), q) - \alpha_j^o \cdot (o_j(q) - \alpha_j^t \cdot t_i) + \alpha_j^{t'} \cdot \min((t_{j,2} - t_{j,1}), (t_{target} - t_{j,1})). \quad (10)$$

where α_*^H, α_*^o are the weights attached by agents to privacy and cost, respectively, α_*^t is the weight attached to the reputation of other agents, $\alpha_*^{t'}$ is the weight attached to one's own reputation, $t_{i,0}$ is $\mathcal{P}_i$'s reputation before starting the protocol, $t_{i,1}$ is $\mathcal{P}_i$'s reputation after requesting and obtaining $\mathcal{P}_j$ to submit q, $t_{j,1}$ is $\mathcal{P}_j$'s reputation before accepting to submit q on $\mathcal{P}_i$'s behalf and $t_{j,2}$ is $\mathcal{P}_j$'s reputation after submitting q. Note that $\mathcal{P}_i$ and $\mathcal{P}_j$ can accurately anticipate $t_{i,1}$ and $t_{j,2}$, respectively (they do not need to use estimates); this is because in our case study reputation is objective, based on help provided and received.

Considering reputations as atomic utilities as in Expressions (9) and (10) further reinforces the co-utility of the protocol, because now responder agents are additionally motivated to collaborate (in order to increase their own reputations) and initiator agents need to make sure they have high enough reputations to afford their queries to be submitted by other peers (which decreases their reputation). Moreover, as discussed in Section 6, viewing reputations as utilities also contributes to a fair allocation of query submission costs among the peers.

7.3. Other benefits of co-utile reputation enforcement

Co-utile protocols are viable given the availability of agents with appropriate types/preferences. In this section we illustrate in our case study how the use of reputation makes the protocol realistically self-enforcing by widening the range of agent types and situations in which the query submission protocol remains co-utile.

First, let us consider the case in which a new agent with an empty profile (i.e., she has not submitted any query so far to the WSE) joins the network. Recall that an empty profile has maximum privacy because it does not disclose any interest of the agent. Even neglecting the cost associated to the query submission, such a newcomer will always prefer to rely on others to submit her queries and will refuse submitting queries from others because doing so will decrease her privacy. In plain words, the newcomer becomes a rational "free rider".

This is an endemic problem of many P2P co-utile protocols (such as file sharing) that can be fixed using reputation. Like in EigenTrust, in our distributed reputation mechanism newcomers have zero global reputation, because no one has interacted with them and the distributed reputation calculation is based on the local reputations, which are zero for newcomers. Thus, in the reputation-based protocol discussed in this case study, responder agents will systematically refuse requests for query submission from free riders. Given that the functionality (i.e., to have the query answered) is likely to dominate the other utilities, the free-riding agent has only two options left:

- She submits her query herself. This action, in turn, will create a non-empty profile for the agent and a sharp decline of the profile's privacy because the agent is now completely exposed to the WSE. As a result, the agent will become highly motivated to contribute to the system by submitting other agents' queries in order to flatten her profile again.
- She accepts a query request from another agent. This results both in a cost and in a decrease of her privacy, even though her profile gives a distorted view of her interests towards the WSE because the query is unlikely to match her true interests. However, this action will also increase the agent's reputation, who will now be able to find another peer willing to submit her query.

In both cases, newcomers with empty profiles become collaborative members of the network in a self-enforcing way. Note that newcomers with non-empty profiles (i.e., who have already submitted queries before joining the network) are not problematic at all because they are already motivated to accept queries from other peers to flatten their profiles.

The above discussion can be extended to an extreme scenario in which all agents join the system at once with empty profiles and zero global reputations. One might think that this setting brings the system to a standstill in which any request for query submission by any agent is refused and agents do not want to submit their own queries. However, it is easy to see that, thanks to functionality dominating the other utilities, agents will end up submitting their own queries; in turn, this will instantly motivate them to submit queries from peers, which will also increase their reputations and the chances that other agents accept their requests.

A different scenario which might also seem standstill-prone occurs when agents achieve a non-empty but flat profile (and probably a high reputation) thanks to the number of query requests they have satisfied; at this point, they might be tempted to stop collaborating. In the unlikely case that all agents reach such a state, they will be unable to find peers willing to submit their queries, regardless of the reputations they have; as a result, they will be forced to submit their own new queries themselves, which in turn will unflatten their profiles and make them again willing to collaborate. If only some of the agents reach this "too happy" state, their reputation will decrease if they ask other peers to submit their new queries; when their reputation becomes too low, they will be unable to find any other peer willing to submit their queries, which will force them to submit them themselves; this will make them again willing to collaborate.

Finally, the fact that agents may also consider their own reputations as atomic utilities has two virtuous effects. On the one hand, since the responder increases his reputation as a result of the submission whereas the initiator decreases her reputation by the same magnitude, the reputation management guides the system towards a state in which most of the agents have a similar –target– reputation, which corresponds to a *fair* allocation of costs/negative payoffs. On the other hand, in the basic protocol, initiator agents randomly

chose potential responders, who may or may not fulfill their requests according to their –secret– privacy needs (profile). With the reputation mechanism, initiator agents may also take into account the reputation of the potential responders so that, instead of choosing them randomly, they may choose agents with zero or low reputation, because (for an equal secret profile) these are more motivated to help in order to increase their own reputations (and thus be able to get help from others). In this way, the initiator increases the chances of finding a helpful responder.

7.4. Implementation and experiments

In this section, we empirically study the behavior of the privacy-preserving query submission protocol and the influence of the reputation mechanism to foster collaboration between agents.

The agent profiles w.r.t. the WSE have been characterized by following the state of the art in user profiling [26]: profiles are represented as vectors of normalized weights, each one quantifying the relative interest of the agent in a certain topic. Each topic weight represents the aggregation of the semantics of the user's queries that fall into the topic, which is quantified by means of linguistic and semantic analyses (see details in [26]).

The simulations have been carried out with 900 agents. Each one has a set of queries for which she wishes to get an answer from the WSE. These queries were randomly picked from the query logs of real users available in the AOL data set [2], which consists of logs of the actual queries performed by users of the AOL's WSE during 3 months in 2006. In total, we compiled 20,914 individual queries to be performed by the agents through the life cycle of the system. In our simulations, agents iteratively select a query from their query logs and decide whether or not to follow the anonymous submission protocol to retrieve the result from the WSE according to their states (current profiles of submitted queries) and types (utility functions). When applicable, global reputations are computed after each query submission.

We have set up the initial profile of each agent to follow the distribution of topics defined by the query log associated to each agent; we do this so that the queries initially submitted by each agent reflect her inherent interests. With this configuration, the average entropy of the initial profile of the agents is 2.15.

We evaluated the behavior of the system in the different scenarios discussed in Sections 7.1 and 7.2, as follows:

- *S1*: In the simplest scenario, neither costs nor reputations are considered. Agents only care about the functionality of the query and their own privacy. Their utility functions are those of Expressions (4) and (5). Utility coefficients have been set to make functionality dominate, so that queries will be *always* submitted, either by the initiators or by other agents, regardless of the privacy gain/loss.

- *S2*: In the second scenario, the cost of submitting a query is considered. The utility functions of agents follow Expressions (6) and (7). The cost of

a query $o(q)$ for the submitting agent is quantified as $+1$. We performed several executions by varying the relative importance (weighting coefficient α^o) that agents attach to the submission cost in their utility functions.

- *S3*: In the third scenario, both costs and agent reputations (measured as the difference between costs borne and caused) have been considered. In this case, responder agents have the utility of Expression (8). Given that reputations are normalized in the [0..1] range and reputation values of all peers add to 1, to properly aggregate costs and reputations, the latter have been de-normalized by setting $\alpha_j^t = \text{number\_of\_peers} = 900$. Again, we performed several simulations by modifying the relative importance of the cost (cost/reputation difference for responder agents).

- *S4*: In the fourth scenario, agents also consider how their own reputation would be affected by the actions they can take (i.e., query forwarding or submission). Initiator agents follow Expression (9) and responder agents follow Expression (10). The target reputation value has been set to the average reputation ($t_{target} = 1/\text{number\_of\_peers} = 1/900$), so that reputations will tend to be balanced and, thus, costs will be fairly allocated. To properly aggregate costs and reputations, we have set the coefficient $\alpha_*^{t'} = \text{number\_of\_peers} \cdot \alpha_*^o$ for each simulation. In this manner, the difference of reputations is de-normalized (so that it can be aggregated with cost values) and given the same weight as agent reputations in scenario *S3*.

The system behavior has been evaluated according to the following aspects:

- Number of queries that have been submitted by following the co-utile protocol. To put this value in context, we also compute the ratio between this number of queries and the total number of queries that initiator agents decide to forward according to their types, that is, the number of queries that potentially can be submitted in a co-utile way. A rate around 1 would be desirable, since it means that the protocol remains self-enforcing for most situations and agent types.

- The standard deviation of agents' reputations. Since our reputation measures the difference between the cost incurred by an agent when submitting other agents' queries and the cost caused to other agents who agree to submit the agent's own queries, a low deviation from the average reputation (which by definition is 1/number_of_peers) indicates a fair allocation of costs resulting from agent collaboration across the network.

- The privacy of agents' profiles, measured as the average entropy of their profiles at the end of each simulation.

For Scenario *S1*, Figure 1 shows the evolution (on average over all the agents) of the agents' privacy (Y-axis), as a function of the number of queries that have been submitted so far (X-axis). The horizontal line at the top shows the privacy

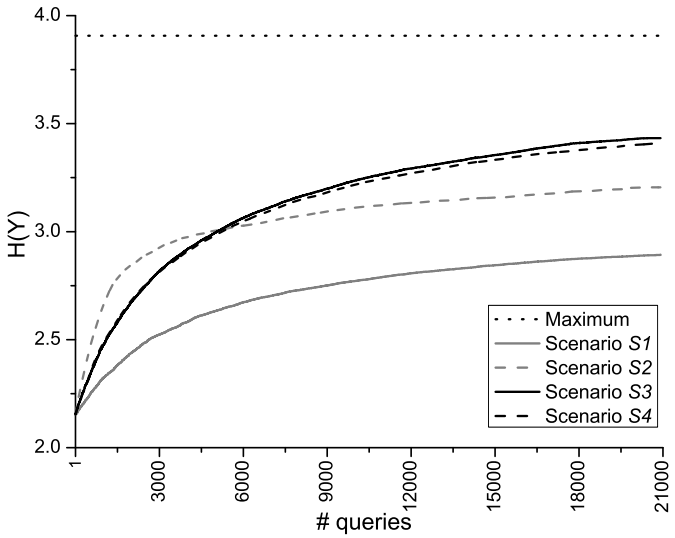

Figure 1: Evolution of privacy (agents' profile entropy) for all scenarios with a cost coefficient of 0.25 (for *S2*, *S3*, *S4*)

upper bound (i.e., the maximum entropy of the profiles when all agents achieve a perfectly flat distribution of the 15 topic weights considered in their profiles, which is $\log_2 15 = 3.9$).

We can see that the agent profiles become more private w.r.t. the WSE at each new iteration of the protocol. The entropy $H(Y)$ grows logarithmically and becomes quite flat after 10,000 queries. It is important to note that, in practice, the limited variability of the queries (whose topics are not evenly distributed among the 15 possible topics) makes it difficult to achieve this theoretical upper bound (in which all agent profiles are perfectly flat). The average entropy of the agents' profiles at the end of the simulation is 2.9.

In this scenario, initiator agents preferred to forward 11,544 queries (from the total 20,914 queries), because submitting them directly would have decreased their privacy. From these, 10,806 queries (93.61%) were submitted by following the co-utile protocol; that is, in less than 7% of the cases, no other agent was willing to satisfy the initiator's request and the initiator was forced to submit the query herself (because the functionality utility dominates). This illustrates how, in absence of costs, the game is naturally co-utile for most agent types.

For Scenario *S2*, Table 1 shows the simulation results obtained by varying the relative importance given to the cost of performing a query to the WSE (α_i^o coefficient in Expression (6) for initiators and α_j^o coefficient in Expression (7) for responders).

We can clearly see how, the more the importance the agents attach to the cost of the queries, the more they tend to refuse submitting queries from other peers (even if refusal implies missing the chance of a privacy gain). As a result, initiators are forced to ask more peers and, if none of the latter is willing to

Table 1: Scenario *S2*: number and proportion of queries submitted to the WSE following the co-utile protocol, and average profile entropy (privacy) at the end of the simulation, for different cost coefficients.

cost coefficient	co-utile queries (proportion)	average profile entropy
4	0 (0)	2.15
1	398 (0.019)	2.62
0.25	2,930 (0.14)	3.20
0.063	8,788 (0.42)	3.47
0.016	15,816 (0.83)	3.35
0.004	14,341 (0.93)	3.05
0.001	11,910 (0.93)	2.95

satisfy their requests, initiators will end up submitting their queries themselves. This makes the system not co-utile for many agent types. Only when agents give a small importance to the cost (below 0.016), they follow the protocol in most cases (i.e., for more than 80% of the queries to be submitted). We can also see that, even though the proportion of co-utile queries monotonically increases as the cost coefficient decreases, the actual number of co-utile queries starts to decrease for costs coefficients below 0.016. Indeed, when initiator agents give very little importance to the query costs, they tend to submit themselves the queries that do not harm their privacy, rather than using the protocol just to save submission costs.

The privacy of agents is also significantly affected by the lack of co-utility: profile entropies are hardly improved (from the initial average entropy of 2.15) when cost coefficients are high enough (from 1 to 4) to prevent the system from being co-utile in most cases. Interestingly, profile entropies reach a high value (around 3.2) when just 14% of the queries are submitted by following the co-utile protocol (see the privacy evolution of this case in Figure 1). This shows how even submitting a small amount of *privacy-preserving* queries from other peers has a very positive flattening effect on the responder's profile. In fact, due to the large weight given by the agents to the query costs, the queries they actually submit on behalf of other agents *have to* provide large privacy gains that compensate their submission cost (see Expression (7)).

Table 2 shows the results for Scenario *S3*, in which reputation is also considered by responders as a payoff.

In comparison with the previous scenario, Scenario *S3* results in a significantly greater number of co-utile query submissions for the same coefficients of cost (cost-reputation difference, in this case). The co-utile query proportion among the forwarded queries, in fact, reaches and maintains a value above 0.9 for cost coefficients below 0.063; this is is comparable to the proportion of Scenario *S1*, in which costs were not considered at all. The actual number of queries submitted in a co-utile way is also higher than in Scenario S2, especially for high cost coefficients. Reputation deviations tend to stay steady for cost coefficients

Table 2: Scenario *S3*: number and proportion of queries submitted to the WSE following the co-utile protocol, average profile entropy (privacy), and standard deviation of agents' reputations at the end of the simulation, for different cost coefficients.

cost coefficient	co-utile queries (proportion)	average profile entropy	agents' reputation std. deviation
4	7,898 (0.37)	2.80	0.00047
1	8,252 (0.39)	3.16	0.00044
0.25	13,501 (0.64)	3.44	0.00038
0.063	18,986 (0.92)	3.34	0.00038
0.016	18,257 (0.97)	3.10	0.00037
0.004	14,611 (0.95)	2.98	0.00037
0.001	10,899 (0.93)	2.90	0.00037

Table 3: Scenario *S4*: number and proportion of queries submitted to the WSE following the co-utile protocol, average profile entropy (privacy), and standard deviation of agents' reputations at the end of the simulation, for different cost coefficients.

cost coefficient	co-utile queries (proportion)	average profile entropy	agents' reputation std. deviation
4	14,968 (0.71)	3.11	0.00017
1	16,201 (0.77)	3.30	0.00017
0.25	17,258 (0.82)	3.41	0.00021
0.063	19,671 (0.95)	3.32	0.00026
0.016	18,307 (0.96)	3.13	0.00028
0.004	14,685 (0.95)	3.01	0.00028
0.001	12,090 (0.94)	2.93	0.00029

below 1.

Finally, Table 3 shows the results for Scenario *S4*, in which the impact of the various available actions on the agent's own reputation is also considered as an atomic utility.

In this case, the interest of the agents to increase their own reputation makes them more motivated to accept query requests from other peers, in spite of query costs or initiators' low reputations. As a result, the system becomes co-utile in most cases and for any cost coefficient values. In fact, the proportions of co-utile queries are quite similar to the ones of scenario *S1*, in which no costs were considered, and the actual numbers of queries submitted in a co-utile way are greater than in any of the previous scenarios.

Moreover, in comparison with Scenario *S3*, in Scenario *S4* we obtain lower standard deviations for the reputation values. Agents are motivated to reach the target reputation but not to exceed it. As a consequence, agents with high enough reputations refuse query requests more frequently than agents with low reputations, which yields a fairer allocation of query costs among the peers.

Finally, it is important to note the beneficial effect of reputation (as a cost-

neutralizing feature) on the privacy finally achieved by the agents. In comparison with Scenario *S1* (for which we obtained an average profile entropy of 2.9), with the reputation mechanism we were able to reach substantially higher figures for all cost coefficients, as shown in Figure 1. Indeed, due to the incurred costs and reputation differences resulting from forwarding or submitting queries, agents become more selective (in terms of the privacy gains) about which queries they forward or accept. As a consequence, the privacy gains of queries are better allocated among the peers.

8. Conclusions

Co-utility happens when rational selfish agents help each other. Co-utile protocols are not only self-enforcing, but they also provide Pareto-optimal utilities to the participating agents that are strictly greater than the utilities they would obtain by not participating. Hence, it is highly desirable to find such win-win protocols to organize any peer-to-peer interaction.

However, in many situations, negative payoffs (cost, reluctance, fear, etc.) may render mutual help not rational. In this paper we have demonstrated the use of reputation to provide artificial incentives that can compensate negative payoffs, thereby making co-utility still attainable. Specifically, we have adapted and extended the EigenTrust mechanism to obtain a general distributed reputation management protocol that can be applied to a variety of scenarios and reputation needs. The resulting protocol is itself strictly co-utile, robust against a number of attacks and compatible with peer anonymity. In this way, we have shown how we can achieve a virtuous circle by which the reputation mechanism is self-enforcing and, in turn, enables co-utility in protocols that would not be co-utile without reputation, due to negative payoffs. This provides a key tool to design self-enforcing protocols that favor mutual help and boost social welfare even in case the agents' natural interests are inauspicious to this end.

As future work, we plan to apply our co-utile distributed reputation mechanism to other scenarios in the information society (e.g., P2P file sharing) and also in the physical world (e.g., car sharing).

Acknowledgments and disclaimer

Funding by the Templeton World Charity Foundation (grant TWCF0095/AB60 "CO-UTILITY") is gratefully acknowledged. Also, partial support to this work has been received from the Government of Catalonia (ICREA Acadèmia Prize to J. Domingo-Ferrer and grant 2014 SGR 537), the Spanish Government (projects TIN2011-27076-C03-01 "CO-PRIVACY", TIN2014-57364-C2-1-R "SmartGlacis" and TIN2015-70054-REDC) and the European Commission (projects H2020-644024 "CLARUS" and H2020-700540 "CANVAS"). The authors are with the UNESCO Chair in Data Privacy, but the views in this paper are the authors' own and are not necessarily shared by UNESCO.

References

[1] B. Adler, L. de Alfaro. A content-driven reputation system for the Wikipedia. In: Proceedings of the 16th international conference on World Wide Web (WWW), ACM, 2007, pp. 261-270.

[2] AOL Search Data Mirrors. Available at *http://gregsadetsky.com/aol-data/*. Last accessed: Oct. 14, 2015.

[3] BlaBlaCar, *http://www.blablacar.com/*. Last accessed: Oct. 14, 2015

[4] J. Carverlee, L. Liu, S. Webb. The SocialTrust framework for trusted social information management: architecture and algorithms. Information Sciences 180(1) (2010) 95-112.

[5] G. Chalkiadakis, E. Elkind, M. Wooldridge. Cooperative game theory: basic concepts and computational challenges. IEEE Intelligent Systems 27(3) (2012) 86-90.

[6] G. Chalkiadakis, E. Elkind, M. Wooldridge. Computational Aspects of Cooperative Game Theory, Morgan & Claypool, 2011.

[7] J. Domingo-Ferrer, Ú. González-Nicolás. Rational behavior in peer-to-peer profile obfuscation for anonymous keyword search. Information Sciences 185(1) (2012) 191-204.

[8] J. Domingo-Ferrer, D. Sánchez, J. Soria-Comas. Co-utilit - self-enforcing collaborative protocols with mutual help. Progress in Artificial Intelligence 5(2) (2016) 105-110.

[9] J. Domingo-Ferrer, J. Soria-Comas, O. Ciobotaru. Co-utility: self-enforcing protocols without coordination mechanisms. In: Proceedings of the 5th International Conference on Industrial Engineering and Operations Management-IEOM 2015, IEEE, 2015, pp. 1-7.

[10] M. Feldman, C. Padimitriou, J. Chuang, I. Stoica. Free-riding and whitewashing in peer-to-peer systems. IEEE Journal on Selected Areas in Communications 24(5) (2006) 1010-1019.

[11] R. Guha, R. Kumar, P. Raghavan, A. Tomkins. Propagation of trust and distrust. In: Proceedings of the 13th International Conference on the World Wide Web, ACM, 2004, pp. 403-412.

[12] K. Hoffman, D. Zage, C. Nita-Rotaru. A survey of attack and defense techniques for reputation systems. ACM Computing Surveys 42(1) (2009) art. no. 1.

[13] L. Hurwicz. Optimality and informational efficiency in resource allocation processes. In K. J. Arrow, S. Karlin and P. Suppes (Eds.) *Mathematical Methods in the Social Sciences*, Stanford University Press, pp. 27-46. 1960.

[14] L. Hurwicz. On Informationally Decentralized Systems. In C. B. McGuire and R. Radner (Eds.) *Decision and Organization: A volume in Honor of Jacob Marschak*, pp. 297-336, 1972

[15] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th International Conference on World Wide Web, ACM, 2003, pp. 640-651.

[16] Y.A. Kim, R. Phalak. A trust prediction framework in rating-based experience sharing social networks without a Web of Trust. Information Sciences 191 (2012) 128-145.

[17] B. Lagesse. Analytical evaluation of P2P reputation systems. International Journal of Communication Networks and Distributed Systems 9(1-2) (2012) 82-96.

[18] B. N. Levine, C. Shields, N. B. Margolin. A survey of solutions to the Sybil attack. Tech report 2006-052, University of Massachusetts Amherst, Amherst, MA, 2006.

[19] K. Leyton-Brown, Y. Shoham. Essentials of Game Theory: A Concise, Multidisciplinary Introduction, Morgan & Claypool, 2008.

[20] S. Marti, H.Garcia-Molina. Identity crisis: anonymity vs reputation in P2P systems. In: Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P 2003), 2003, 134-141.

[21] S. Marti, H.Garcia-Molina. Taxonomy of trust: categorizing P2P reputation systems. Computer Networks 50(4) (2006) 472-484.

[22] M. Osborne, A. Rubinstein. A Course in Game Theory, MIT Press, 1994.

[23] J.A. Pouwelse, P. Garbacki, D.H.J. Epema, H.J. Sips. The Bittorrent P2P file-sharing system: measurements and analysis. In: 4th International Workshop on Peer-To-Peer Systems, LNCS 3640, Springer, 2005, 205-216.

[24] A. Singh, L. Liu. TrustMe: anonymous management of trust relationships in decentralized P2P systems. In: Proceedings of the Third International Conference on Peer-to-Peer Computing (P2P 2003), 2003, 142-149.

[25] M. Srivatsa, L. Xiong, L. Liu. TrustGuard: countering vulnerabilities in reputation management for decentralized overlay networks. In: Proceedings of the 14th International Conference on the World Wide Web, ACM, 2005, 422-431.

[26] A. Viejo, D. Sánchez, J. Castellà-Roca. Preventing automatic user profiling in Web 2.0 applications. Knowledge-based Systems, 36 (2012) 191-205.

[27] K. Walsh, E.G. Sirer. Experience with and object reputation system for peer-to-peer filesharing. In: Symposium on Networked System Design and Implementation (NSDI), 2006.

[28] R. Zhou, K. Hwang. PowerTrust: a robust and scalable reputation system for trusted peer-to-peer computing. IEEE Transactions on Parallel and Distributed Systems 18(4) (2007) 460-473.

[29] R. Zhou, K. Hwang, M. Cai. GossipTrust for fast reputation aggregation in peer-to-peer networks. IEEE Transactions on Knowledge and Data Engineering 20(9) (2008) 1282-1295.