Using Social Networks to Distort Users' Profiles Generated by Web Search Engines

Alexandre Viejo [a,b], Jordi Castellà-Roca [a]

[a] *Departament d'Enginyeria Informàtica i Matemàtiques, UNESCO Chair in Data Privacy, Universitat Rovira i Virgili, Av. Països Catalans 26, E-43007 Tarragona, Spain*
E-mail: {alexandre.viejo,jordi.castella}@urv.cat

[b] *Institute of Information Systems, Humboldt-Universität zu Berlin, Spandauer Str. 1, D-10178 Berlin, Germany*
E-mail: alexandre.viejo@wiwi.hu-berlin.de

Abstract

The Internet is one of the most important sources of knowledge in the present time. It offers a huge volume of information which grows dramatically every day. Web search engines (e.g. Google, Yahoo...) are widely used to find specific data among that information. However, these useful tools also represent a privacy threat for the users: the web search engines profile them by storing and analyzing all the searches that they have previously submitted. To address this privacy threat, current solutions propose new mechanisms that introduce a high cost in terms of computation and communication. In this paper, we propose a new scheme designed to protect the privacy of the users from a web search engine that tries to profile them. Our system uses social networks to provide a distorted user profile to the web search engine. The proposed protocol submits standard queries to the web search engine; thus it does not require any change in the server side. In addition to that, this scheme does not require the server to collaborate with the users. Our protocol improves the existing solutions in terms of query delay. Besides, the distorted profiles still allow the users to get a proper service from the web search engines.

Key words: Privacy, Private information retrieval, Social networks, Web search

1 Introduction

Nowadays, there is a huge volume of information available on the Internet. More importantly, that information grows every day and its limit cannot be envisaged.

Web search engines (WSE) — e.g. Google, Yahoo or Microsoft Live Search — are widely used to find specific data among that information. The overwhelming amount of data that they gather is usually provided to the users using several result pages (web pages containing links to the resulting data). [1] states that 68% of the users of a web search engine click a search result within the first page of results. Also, 92% of the users click a result within the first three pages of search results. Therefore, in order to provide a better user experience, web search engines should put in the first result pages the links that are more interesting for the users.

Nevertheless, it is not easy to know the users' interests. An example of that happens when looking for the word "Mercury". This term can refer to the planet Mercury or to an element in the periodic table. The concept *disambiguation* represents the process of identifying the correct sense when a certain word has different ones. The disambiguation process requires the knowledge of: (i) the interests of the user; or (ii) the query context. Both can be deduced from the user's profile, which is built from her previous searches. For example, if a certain user has searched "solar system" before "Mercury", the web search engine will assume that "Mercury" refers to the planet Mercury and not to the element in the periodic table. According to that, the web search engine will put the results that correspond to the planet Mercury in the first pages.

How the users of web search engines are profiled is a topic which has been widely addressed in the literature. In [2], the authors use the browsing history. [3] proposes the use of click-though data. [4] introduces the use of web communities for this purpose. A client side application which stores users' interests is presented in [5]. However, the most successful approach for the web search engines is the use of the queries previously submitted by the users [4,6]. This mechanism effectively profiles the users and it does not require their collaboration.

Even though the use of profiles improve the users' experience, they contain information that can be considered private and personal. If a certain user has searched for a certain place, it can be inferred that she lives there. If she looks for a certain disease, it can be deduced that she (or someone close to her) suffers that disease. This situation poses a serious threat to the users. The AOL scandal, where 20 million queries made by 658000 users were publicly disclosed [7], proves that the web search engines cannot protect properly the privacy of their users. Therefore, the users should use privacy-preserving mechanisms to prevent the web search engines from profiling them in a *detailed* way. Note that profiles are needed in order to provide an efficient service to the users. Thus, there is a trade-off between the privacy level achieved and the quality of the service. If the user desires a high degree of privacy, she will probably receive a deficient service. If the user desires an accurate service, her privacy will probably be jeopardized.

In the literature, the process of improving the accuracy of the web search engines by profiling users receives the name of *personalized search (PS)* or *personalized web search (PWS)* [8–12]. As explained above, these mechanisms put at risk the privacy of the users.

In [11], four levels of privacy protection are defined and analyzed: *pseudo identity*, *group identity*, *no identity* and *no personal information*.

In the *pseudo identity* level, the user identity is replaced by a pseudo-identity which contains less identifiable information. Nevertheless, this does not prevent the WSE from creating a profile associated to the pseudo-identity. This profile contains sensitive information that can be used to identify the real user who hides behind the pseudo-identity. For example, when AOL released its search engine log in 2006, they replaced IP addresses with pseudo-identities [7]. In that circumstance, this level of protection was proved to be insufficient.

The second level of privacy corresponds to the *group identity* level. In this case, a group of users share a single identity. Therefore, the WSE is only able to build a group profile. It cannot profile single users. This mechanism improves the privacy level achieved by users. However, the use of a group profile instead of individual profiles reduces the effectiveness of the service.

There are three ways to implement this level of privacy: (i) using a proxy to construct the group; (ii) using an obfuscation mechanism like submitting random queries; and (iii) sending queries which have been generated by other users.

One shortcoming of the proxy approach is that then the proxy can build the individual profiles. Regarding the use of obfuscation methods, submitting random queries misrepresent the profiles of the users. There are two works based on this solution that have been implemented in the Firefox web browser: TrackMeNot [13,14] and GooPIR [15]. The authors in [11] state that this approach can be considered as a way to get k-anonymity [1] [16]. Both TrackMeNot and GooPIR are fast because they do not create groups. However, they have the following drawbacks:

- TrackMeNot submits random queries to the web search engines when the users' activity is low. This is done to prevent the system from affecting the users' normal work. Nevertheless, sending fake queries increases the network traffic and overloads the WSEs. Therefore, this scheme protects the privacy of the users, but it reduces the network and WSEs' performance. In addition

[1] See [16] for a definition of k-anonymity.

to that, this behaviour introduces a serious privacy threat: for each user, the WSE is able to divide all her queries depending on whether or not they have been submitted during working hours (according to the time-zone of the user). Probably, all the queries submitted out of the working hours have been sent by TrackMeNot. The period of time between two different queries can also be used to the same purpose: it can be assumed that when the users are working, they do not submit only one query but several in a short period of time. This gap between queries can be used to deduce whether or not a certain query has been submitted by TrackMeNot.

- GooPIR submits fake words to the WSE together with the authentic one. This behaviour obfuscates the user's profile because the WSE cannot know which words are fake and which are not. This proposal introduces a large overhead to neither the network nor the WSE. Nonetheless, GooPIR uses a Thesaurus in order to decide which words can be added to the search. According to that, GooPIR can only submit words. Full sentences are not addressed (note that sentences cannot be formed by random words).

As explained previously, the last approach to implement the *group identity* level is based on users submitting queries generated by other users. There are two different proposals in the literature that follow this approach: [17,18].

The system proposed in [17] uses memory sectors which are shared by a group of users. These users use the shared memory to store and read the queries and their answers. There is no connection between the users. Queries and answers are encrypted in order to provide confidentiality. This proposal does not require a trusted third party to create the groups or generate the cryptographic material. Instead of that, a simple wiki-like collaborative environment can be used to implement a shared memory sector. An onion routing protocol is used to preserve the privacy of the users that access this environment.

This scheme has the following drawbacks:

- It should be capable of managing a high volume of information. However, the memory-space requirements have not been studied by the authors.
- Users must scan their shared memory sectors at regular intervals. This requirement introduces a significant overhead to the network.
- The best response time achieved by this proposal is 5.84 seconds. However, the authors do not include in this time the network time. According to that, the final response time is expected to be clearly above 5.84 seconds but the exact value is not specified.

In [18], the Useless User Profile (UUP) protocol is proposed. The main idea of this scheme is that each user who wants to submit a query will not send her own query but a query of another user instead. Using this approach, the web search engine cannot generate a real profile of a certain individual. Regarding

privacy concerns, the relevant point is that the users do not know which query belongs to each user. This is achieved using cryptographic tools. The system requires two components: a central node and a client application. The central node listens to client requests. After receiving n requests, it creates a new group and sends the IP addresses and port numbers to each group member. Then, the group members establish network connections between them and start to communicate without the interaction of the central node. This scheme has been tested in real conditions and provides an overhead of 5.2 seconds with a group of three users and a key length of 1024 bits. The shortcomings of this proposal are the following:

- The groups of users must be created. This process introduces a significant delay.
- It requires a large number of users in order to provide an acceptable response time.
- It does not consider the trade-off between the privacy level achieved and the quality of the service.

In the *no identity* level, the identity of the user is not available to the search engine. As a result, the WSE cannot profile the users. Almost all the proposals that fall in this level use an anonymous channel implementation. The *Tor Project* [19] is an example of that. There are several Firefox plugins [20,21] that are based on Tor.

The main drawback of this approach is that the process of submitting a query to the WSE and receiving the answer through an anonymous channel is very time-consuming. The authors in [10] used the anonymous network Tor with paths of length two (note that the default length is three) and submitting a query was, on average, 25 times slower than performing a direct search.

Private Information Retrieval (PIR) schemes also fall in the *no identity* level of privacy. PIR protocols protect the privacy of the users who retrieve information from servers. Thus, a PIR scheme enables the users to retrieve a certain value from a database while the server, which holds the database, gets no knowledge about the data requested by the users.

The first PIR protocol was designed by Chor, Goldreich, Kushilevitz and Sudan [22,23] . This scheme is based on several servers holding the same database. The authors assume that those servers cannot communicate between them. This proposal cannot be used in scenarios with only one server (single-database PIR). Web search engines are an example of this type of scenario.

Single-database PIR schemes can be found in [24] and [25]. Even though these proposals are suitable for WSEs, they have important shortcomings:

- The database is usually modeled like a vector. In that case, the user wants to retrieve the value stored in the i-th position of that vector. Both assumptions are not realistic because the database of the WSE is not a vector and the user does not know where the WSE stores the information.
- PIR schemes require that the WSE collaborates with the users. The WSE has no motivation to protect the privacy of the users. In addition to that, these schemes increase the computational and communication cost of the WSE.

In the *no personal information* level, the identity of the user and the description of the data she desires are not available to the search engine. This level provides the highest privacy protection to the users. Nevertheless, the computational and communication costs required by these mechanisms make them unaffordable in practice [11].

Note that, in addition to all the schemes explained above, there is a straightforward way to provide anonymity to the users who use WSEs: they can access to the Internet using a dynamic IP address and a controlled/clean web browser without cookies. However, this approach has the following drawbacks:

- The renewal policy of the dynamic IP address is not controlled by the user but the network operator. This operator can always give the same IP address to the same Media Access Control (MAC) address.
- Certain users require static IP addresses.
- A browser without cookies loses its usability in a high number of web applications. This situation may not be affordable for certain users.

1.2 Contribution and plan of this paper

In this paper we propose a new scheme designed to protect the privacy of the users from a web search engine that tries to profile them. Our system provides a distorted user profile to the web search engine. The proposed protocol submits standard queries to the web search engine, thus it does not require any change in the server side. In addition to that, this scheme does not require the server to collaborate with the users.

Our proposal is based on the two following assumptions: (i) the proliferation of home computers equipped with flat-rate broadband connection to the Internet. This implies that computers can be permanently connected to the network; and (ii) the gradual introduction of social networks.

Section 2 introduces the background and the assumptions used throughout the paper. In particular, it details the scenario and the privacy requirements.

In Section 3 we describe our new protocol. In Section 4 we provide simulation results of our scheme and we discuss them. In Section 5 we analyze the behaviour of our protocol when dealing with the adversaries that we envisage. Section 6 discusses how our proposal can be deployed in a real environment. Finally, Section 7 reports some concluding remarks.

2 System model

2.1 Entities and architecture

In our scenario, the following entities are present:

- *Users.* They are the individuals who submit queries to the web search engine. Their motivation is to protect their own privacy.
- *Web search engine.* This is the server that holds the database. It can be Google, Yahoo or Microsoft Live Search, among others. As explained previously, this entity has no motivation to preserve the privacy of its users.

Our scheme requires the users to be organized in *social networks*. A social network is a community of users where each one can publish and share information and services. Specifically, we follow the social network concept explained in [26]. This approach does not require the existence of a central node. Thus, users are connected directly between them or by means of other users who act as intermediaries. In addition to that, these social networks are not open to examination (a user only knows her direct connections in the network).

In Section 6, we give details about how the social network that we envisage is constructed and managed.

2.2 Our proposal in a nutshell

The proposed protocol preserves the privacy of the individuals who deal with a web search engine. In order to do that, it exploits the existence of neighbourhoods of on-line users (social networks). In this way, a user U generates queries and she can submit them directly to the WSE or she can forward them to her neighbours in the social network (a neighbour represents a direct relation in the social network). A neighbour who receives a query can submit it to the WSE or she can forward it again to one of her own neighbours. Note that a query is forwarded between users until someone submits it to the web search engine.

Let us consider a group G formed by U and her neighbours. If the queries generated by all the users in G are evenly distributed among G, the profile of U is distorted with queries that do not belong to her. Therefore, U's profile cannot be linked to U by the WSE. That implies that her privacy is preserved.

Our scheme has the following two objectives:

- It must be usable in practice. The neighbourhoods (groups) are already created, hence the proposed system does not create groups for submitting queries. This represents a significant time reduction in comparison with other proposals in the literature. Also, anonymous channels (like Tor [19]) are not used. As explained previously, they are 25 times slower than a direct connection.
- The WSE must not be able to obtain an accurate profile of the users. However, profiles should be still usable (to a certain level) in order to provide a proper service. If the proposed protocol is being used, the WSE obtains a profile of each user. Nevertheless, this profile is not a detailed one. It contains queries from all the members of the neighbourhood. Our scheme assumes that a certain user shares similar interests (hobbies) with her direct neighbours in the social network. Usually, two individuals become "friends" because they have some common interests. "Friendship" is virtually represented in the social network with a direct connection between these users (they are neighbours).

  According to that, it can be assumed that a profile that has been distorted with queries from neighbours is still usable (to a certain level) by the WSE.

In addition to that, to prevent neighbours from abusing the system (i.e. they never submit another user's query to the WSE), the proposed scheme uses a reward mechanism. Users who do not cooperate are eliminated from the system.

2.3 Adversary model

We identify three different adversaries in our scheme:

- *Web search engine.* This entity is the main adversary in our scenario. It profiles users for its own purposes. The web search engine knows who (IP address, cookies...) submits each query. That information is stored during a certain period of time. For instance, Google stores it for 9 months [27].
- *Dishonest user.* Users collaborate in submitting queries from other individuals to the web search engine. A dishonest user can use her role in the protocol to profile other users. A dishonest user knows who has sent a certain query to her.
- *Selfish user.* A selfish user does not follow the proposed protocol. She sys-

tematically uses the others to submit their own queries to the web search engine. However, she does not accept queries from the others. This is not an intentional attack against the users of the system. Nevertheless, this behaviour prevents the queries from being evenly distributed among the group and it can jeopardize the privacy of the honest users that follow the proposed protocol. According to that, mechanisms to prevent users from behaving in that way should be provided.

2.4 Privacy requirements

When a user U submits a query q to the WSE, the WSE obtains some information id about U. This information can be an IP address or a cookie, and it can also be a combination of these items. Let $ID = \{0,1\}^*$ be the set of strings of information which are obtained by the WSE from each query q_i. Also, let the tuple (q, id) be the *query context*.

The WSE can divide the tuples into groups depending on the id. Then, it can create a list $l_{id} = \{q_1, \ldots, q_r\}$ which contains all the queries that share the same id. Finally, the WSE can infer that all these queries have been submitted by the same user U.

Definition 1 (**Identification**). *Let $G = \{U_1, \ldots, U_n\}$ be the group of users of a WSE and $ID = \{id_i\}^*$ be the set of strings which contains the information about those users. A certain user U_j can be identified as the one who has submitted a certain query q if there is an algorithm that, given a tuple (q, id_i), gives U_j as output with a probability of $1 - \epsilon$ (where epsilon is negligible).*

As explained previously, the WSE can get different types of information from the users. For example, the WSE can get a tuple (q, id_1, id_2) where id_1 is an IP address and id_2 is a user's cookie. Let us assume that the WSE owns a list of queries $l_{id_1} = \{q_1, \ldots, q_l\}$ linked with id_1 and a a list of queries $l_{id_2} = \{q'_1, \ldots, q'_m\}$ linked with id_2. The WSE considers that both informations id_1 and id_2 belong to the same user U, hence both lists of queries can be aggregated into a single one.

Definition 2 (**User context**). *Each user U has a user context C which is formed by different strings containing the information about the user (id). These ids are gathered by the WSE from the query contexts (q, id_i), According to that, $C = \{id_1, \ldots, id_t\}$.*

We can assume that the WSE gets two tuples (q_i, id_1, id_2) and (q_j, id_3, id_2), where id_1 is an IP address, id_2 is a cookie, and id_3 is another IP address. The WSE can aggregate these ids using the cookie (id_2). As a result, the WSE

obtains the following user context $C = \{id_1, id_2, id_3\}$.

Definition 3 (**List of queries**). *A user U can submit the same query q more than once to the WSE. Therefore, the WSE stores the number of times z that a certain user U has submitted the same query q. Formally, this can be reflected using tuples of the form (q_i, z_i). Tuples linked with the same user U are stored in a list $L = \{(q_1, z_1), \ldots, (q_r, z_r)\}$. L is the list of queries of U.*

Definition 4 (**Search history**). *Each user U has a search history H that consists of her user context C and her list of queries L. This can be represented as $H_U = \{C_U, L_U\}$. The WSE knows the web search history of all its users.*

Now, let us consider a group of users $G = \{U_1, \ldots, U_t\}$ who use the proposed scheme. In this case, the list of queries L_i of a user U_i will include her own queries (as usual) and queries which have been generated by other users. In this way, if a user U uses our protocol to submit n times the same query q, it can be assumed that q will appear in L_U and that z will be the number of times that U has submitted q to the WSE by herself. The rest of queries $y = n - z$ will be divided between U's neighbours, between the neighbours of the neighbours and so on. Let $G' = \{U_1, \ldots, U_k\} \subseteq G$ be the group of users (U is not included among them) who have q in their search history. Formally, this can be represented as $(q, y'_j) \in L_{U_i}$ for $\forall U_i \in G'$, where $U \notin G'$.

The WSE cannot know which queries $q_i \in L_{U_j}$ have been generated by U_j and which ones have been generated by U_j's neighbours (or the neighbours of the neighbours). The WSE only knows the number of times z_i that U_j has submitted q_i. The proposed system offers a good protection if z_i does not expose U_j.

Definition 5 (**Exposed user**). *A user U is exposed if the WSE can use z_i to detect with a high probability that a certain query $q_i \in L_U$ has been generated by U, where $(q_i, z_i) \in L_U$.*

We next define the requirements which must be fulfilled in order to guarantee that a certain user U is not exposed:

- *Requirement 1.* Given a certain query q, the WSE can group all the users U_i who satisfy $(q, z_i) \in L_{U_i}$. Then, the WSE orders that group depending on the number of times z_i that each user has submitted q. This is $Y = \{U_1, \ldots, U_k\}$, where $(q, z_i) \in L_{U_i}$, $(q, z_{i-1}) \in U_{i-1}$, and $z_i > z_{i-1}$ for $\forall i \in Y$. Let us consider that U_j was the user who generated q initially, $(q, z_j) \in L_{U_j}$. The maximal uncertainty for the WSE happens when all the users $U_i \in Y$ have submitted q the same number of times. This is $z_i = z_j$ when $i \neq j$, for $\forall i, j \in Y$. This requirement is very strict and it can be difficult to achieve. Therefore, we relax it and we only require z_j to be a value in the range

$z_1 \leq z_j \leq z_k$. This requirement prevents z_j from being out of range. If z_j is out of range, the WSE can easily identify U_j.

- *Requirement 2.* Even if z_i fulfills the first requirement (z_j is in the range), the WSE can identify U_j if her position in the ordered group Y is always the same. According to that, U_j should ideally be situated in the middle of Y but the deviation of this position should be high. A high deviation implies that it is difficult for the WSE to ascertain the exact position of U_j in Y. Thus, the WSE cannot identify U_j.

A user U that fulfills these requirements is concealed into a group of k users and achieves a level of privacy comparable to k-anonymity [16]. This implies that the probability of correctly identifying U is at most $1/k$.

Note that, theoretically, any user who submits q to the WSE is considered in the group of k users. However, from a practical point of view, only the users that submit q a *significant* number of times are considered in the k-group. Usually, these users are the neighbours of U.

3 Our proposal in detail

This section details how a query is submitted to a WSE in our scheme. We also explain two functions which are needed for this operation. These functions estimate the profile exposure level and the trustworthiness of the users of the system.

In Section 3.2, we introduce how our system provides anonymity to the users. Anonymity enables dishonest users to act with total impunity. Therefore, a mechanism which prevents anonymous users from submitting illegal queries is also presented.

Finally, Section 3.3 discusses how this proposal behaves in social networks with trust values between users [26,28,29]. Section 3.4 presents some approaches to prevent intermediate users from tampering with answers.

3.1 Protocol for submitting anonymous queries to a web search engine

Let us imagine a user U_i who wants to submit her query q to a web search engine using our scheme. U_i has k neighbours in the social network. They are noted as $(N_1, \ldots, N_k)$. A protocol execution consists of the following steps:

(1) U_i decides whether to directly submit q to the web search engine or to forward it to a certain neighbour. This decision is made depending on

the current privacy level achieved by U_i. Let $\Psi(U_i, N_1, \ldots, N_k)$ be the function that estimates the profile exposure level of a certain user U_i in respect of her neighbours $\{N_1, \ldots, N_k\}$. Function Ψ returns the user (she can be a neighbour or U_i herself) who should submit q to the web search engine(see Section 3.1.1 for details about Ψ).

If Ψ estimates that U_i must submit q to protect her privacy, U_i submits q to the WSE and the protocol ends here. Otherwise, U_i moves to the next step.

(2) Let us assume that Ψ estimates that the neighbour N_i should submit q to protect the privacy of U_i. Therefore, U_i forwards q to N_i and ask her to accept it. N_i accepts or rejects q depending on the level of selfishness of U_i in respect of N_i. Let $\Upsilon(N_i, U_i)$ be the function that computes the level of selfishness that N_i assigns to U_i. Function Υ returns whether N_i must accept q or not (see Section 3.1.2 for details about Υ).

If N_i accepts q, N_i repeats the same steps followed by U_i in order to decide whether N_i directly submits q to the web search engine or whether she reforwards q to one of her own neighbours.

If N_i rejects q, U_i decides again (using Ψ) who of the remaining candidates (herself and the neighbours who have not been previously asked) should submit q. This process is repeated until someone accepts q. In case that all the neighbours reject q, U_i submits her query by herself.

(3) Let us assume that a certain user U_j accepts q from a neighbour U_l and she submits it to the WSE. Then, U_j receives the answer a and it is forwarded to U_l (note that U_l can be U_i, or a neighbour of U_i, or a neighbour of a neighbour of U_i and so on). This process is repeated until a reaches the user U_i who initially generated q. Then, the protocol ends.

3.1.1 Function Ψ: estimating the profile exposure level

Let us consider a user U who has a query q and a list of neighbours (direct relations in the social network) $\{N_1, \ldots, N_k\}$. U executes Ψ to decide who among $\{U, N_1, \ldots, N_k\}$ should submit q to the web search engine. The purpose of Ψ is to preserve the privacy of U.

According to the privacy requirements introduced in Section 2.4, U's profile is perfectly concealed when all the members of that group have submitted the same number of queries generated by U. Therefore, Ψ evenly distributes all these queries between all the users in the group formed by U and her neighbours.

In order to do that, Ψ estimates the number of queries generated by U that each neighbour has *probably* submitted to the WSE. Let α_i be the estimated number of queries (generated by U) that a certain neighbour N_i has submitted. Note that α_i can only be estimated because U cannot know if N_i has submitted

a query or she has forwarded it to one of her own neighbours. However, U knows the number of queries that N_i has accepted. Let τ_i be this number. Function Ψ computes α_i as follows:

$$\alpha_i = \tau_i / \text{number of neighbours of } N_i \tag{1}$$

Expression (1) requires the number of neighbours of N_i. Since U cannot know how many neighbours a certain user has, it has to be estimated as well. Let β_i be this estimation. We next introduce the methods which have been tested in order to compute β_i:

- *Method 1.* This method assumes that everyone in the social network has the same number of direct connections. According to that, it is considered that each neighbour of U has as many direct connections as U.
- *Method 2.* This method uses the average number of queries that U has tried to forward to a single neighbour N_i. Let θ_i be this value. U also knows the number of direct connections that she has and the number of queries that a certain neighbour N_i has tried to forward to her. Let these values be c and t_i respectively. Then, β_i is computed using a simple cross-multiplication:

$$\beta_i = \frac{t_i \cdot c}{\theta_i}$$

- *Method 3.* This method uses the number of queries t_i that each neighbour N_i has tried to forward to U. All neighbours are put into a list and it is ordered with respect to t_i. The neighbour with the highest t_i (first neighbour in the ordered list) is assumed to have only one connection. The second neighbour in the list is assumed to have two connections and so on.

  The idea beneath this method is that a user with several connections will send fewer queries to each neighbour (U being one of them from this point of view).
- *Method 4.* This method is similar to *Method 3*. But in this case the neighbour with the lowest t_i is assumed to have only one connection.

  The idea beneath this method is that a user with several connections will accept more queries from her own neighbours. As a result of that, she will forward more queries to each neighbour (U being one of them from this point of view).

We have simulated the four mechanisms to check out which one gives a better β_i for every user. In order to do that, we have compared the real number of neighbours of each user U_i with the computed β_i.

These simulations have been performed with a social network of 400 users where each user generated 971 queries (see Section 5 for details about it). For each method, 1000 tests were run and the average of these results were computed. Table 1 shows the probability for each method to match the real

number of neighbours of a certain user (Success probability). It also presents the average error of a certain estimation, i.e. the difference between the real number of neighbours and the value which has been estimated using one of the methods described above.

Table 1
Simulation results of each method

Method	Success probability	Average error
Method 1	11.26 %	2.55
Method 2	10.58 %	2.91
Method 3	14.31 %	2.12
Method 4	12.12 %	2.63

The simulation results show that *Method 3* behaves better than the rest. This method obtains the best success probability and, when it fails, the difference between β_i and the real value is smaller than in the other methods. According to that, *Method 3* is used in the rest of this paper. Note that designing and testing more effective estimation methods is outside the scope of this paper.

3.1.2 Function Υ: evaluating the selfishness

Let us consider a user U who receives a query q from her neighbour N. U executes Υ to decide whether to accept or reject q. The purpose of Υ is to punish the users who behave in a selfish way.

In Section 2.3, we have explained that selfish behaviour is likely to harm the system. Function Υ is a mechanism that prevents users from behaving in that way. We next explain how the function Υ that we have used in our simulations works.

Initially, U assigns to each neighbour a probability p of accepting queries from that source. This p is set to 1.0 (probability of 100%) for each neighbour. For notation purposes, $p_{U,N}$ represents the probability for U to accept a query from N.

Let us assume that U receives a query q from user N. U accepts q with probability $p_{U,N}$:

- If U accepts q, the following happens:
  - N increments her own probability of accepting queries from U:

$$p_{N,U} = p_{N,U} + 2 \cdot \gamma$$

Where γ is a constant which is defined by the system.

· U decrements her probability of accepting queries from N:

$$p_{U,N} = p_{U,N} - \gamma$$

Both operations are done to incentivize users to accept queries from their neighbours. A certain user U_1 who forwards several queries to the same neighbour U_2 without accepting queries in exchange (selfish behaviour) will finally get a $p_{U_2,U_1} = 0$. If it happens with all her neighbours, U_1 will be isolated and forced to submit all her queries to the WSE by her own. An isolated user is able to improve her situation by accepting queries from her neighbours. By decreasing γ for a reject and increasing $2 \cdot \gamma$ for an accept, the protocol punishes the users that systematically reject all queries instead of the users that accept and reject queries in an even way.

- If U rejects q, then N decrements her own probability of accepting queries from U:

$$p_{N,U} = p_{N,U} - \gamma$$

We have simulated our scheme for several values of γ to check out which one better isolates the selfish users of the system.

These simulations have been performed with a social network of 400 users (10% of users were selfish) where each user generated 971 queries (see Section 5 for details about it). For each γ, 1000 tests were run and the average of these results were computed.

Table 2 reflects how γ affects the % of exposed users considering two categories: honest users and selfish users.

Table 2
Simulation results for several values of γ

γ	% exposed sefish users	% exposed honest users
0.00	2.5 %	55.8 %
0.01	97.5 %	46.3 %
0.02	100 %	44.1 %
0.03	100 %	48.0 %
0.04	100 %	48.3 %
0.06	100 %	48.8 %
0.08	100 %	53.0 %
0.10	100 %	69.4 %

The simulation results show that when $\gamma = 0$ (the system does not react

to selfishness), the selfish users are not punished by their behaviour and the honest users suffer the consequences (55.8% of honest users are exposed). This situation improves when the system applies measures against selfishness ($\gamma = 0.01$). In this case, 97.5% of selfish users are exposed. Finally, from $\gamma = 0.03$ onwards, the system starts to punish also the honest users. This situation becomes worse when γ grows. In this way, 69.4% of honest users are exposed when $\gamma = 0.10$.

The best results were achieved with $\gamma = 0.02$. With this value, the system punishes more selfish users and protects more honest users. Therefore, this is the value used in the rest of this paper. Note that designing a better algorithm for evaluating the selfishness of the users is outside the scope of this paper.

3.2 Users' anonymity

In our scheme, we enforce anonymity for users who are not directly related. For example, if a query q is forwarded through a path of users U_1-U_2-U_3-U_4 (let us consider that user U_1 is the real generator of q), this kind of anonymity means that U_4 only knows that q comes from U_3 and she has no knowledge about which users or how many users are behind U_3; the same situation occurs between U_3 and U_2, and so on.

The existence of total anonymity can encourage a dishonest user to send a query which breaks the law. Let us assume that U_1 behaves in that way. Our protocol works without the interaction of the user herself. It is executed by the software running in the computers of the users. Thus, it can be difficult for that software to identify a legal query from an *illegal* one. As a result, user U_4 can finally submit one of these illegal queries without notice its illegal content. Later, this submission can point U_4 as the generator of the query in front of the governmental authorities (e.g. the police). If this situation occurs, U_4 will not be able to prove her innocence and she will be punished by the law. On the other hand, U_1 will receive the answer to her query and she will not be punished for her misbehaviour. Such a situation is really dangerous and must be prevented. Measures against this threat fall into two classes: *a priori* and *a posteriori*.

A priori measures would consist of checking the query before accepting it. This process should be done by a filter that discards queries depending on their content. This filter should be carefully designed in order to not require the final user to perform an excessive amount of work. How this filter can be implemented is outside the scope of this paper and requires further research. Two possible directions to address this issue would be: (i) allow the users to select categories of words to be discarded. These categories would be based

on the ones defined by the Open Directory Project (ODP) [30]; and (ii) use ontologies to process the content of the queries. References [31,32] are examples of the work done in this field.

The use of filters has two main shortcomings: (i) it increases the cost in time; and (ii) it is difficult to guarantee that all illegal queries will be filtered. These two problems are solved using *a posteriori* measures. These methods are based on the use of digital signatures to prove the existence of a query transaction between two users.

Digital signatures are covered by the European Law and they are admissible as evidence in legal proceedings [33]. The European Union adopted a community framework for digital signatures (directive 1999/93/EC)) that has been implemented in various European countries. This directive considers three types of digital signatures. We focus on the *advanced electronic signature* which is based on a qualified certificate and which is created by a secure-signature-creation device (also called a Certification Authority or CA). This type of digital signature has a strong judicial value: it warranties authentication, integrity, confidentiality and non-repudiation. As a consequence, *a posteriori* measures can rely on this kind of signatures to provide legal protection to the users.

Our proposal uses a *liability mechanism* to prove the innocence of the users. This is an *a posteriori* measure that provides *partially* revocable anonymity: the anonymity of the users only vanishes in front of a governmental authority. We next explain this mechanism in detail.

3.2.1 Liability mechanism

Our liability mechanism protects the privacy of the honest users while enabling them to prove to a third party (a governmental authority) that they have not generated a certain query.

To accomplish that, we use certificates as a proof of the query transaction between two users. This method was first introduced in [29]. For example, let us imagine a path U_1-U_2-U_3, where U_1 is the true generator of a certain query q and U_3 is the user who finally submits q to the WSE. U_1 sends the pair $(q, \sigma_{U_1,U_2}(q))$ to U_2, where $\sigma_{U_1,U_2}(q)$ stands for the certificate which proves the transaction of q between U_1 and U_2. This certificate is signed by U_1 using her secret key (SK_{U_1}); the corresponding public key (PK_{U_1}) is assumed to be known and accepted by at least the neighbours of U_1 (in this example user U_2). In turn, U_2 sends $(q, \sigma_{U_2,U_3}(q))$ to U_3.

A certificate for a transaction where user U_i forwards a query q to user U_j is constructed as follows:

$$\sigma_{U_i,U_j}(q) \leftarrow \{q||U_i||U_j||time\_stamp\}_{SK_{U_i}} \quad (2)$$

Expression (2) contains the identity of the two nodes involved in the transaction, the forwarded query and the *time_stamp* which reflects when it was sent. A user U_j who receives a certificate must store it in a safe place. Later, if asked by a third party about the origin of a certain query, U_j can use σ_{U_i,U_j} to prove that she is only an intermediate node.

A certain user, who does not own a certificate which proves its innocence, will be considered guilty immediately. There are two reasons for a user to not to have this certificate:

- *She is the legitimate owner of the illegal query.* Such a user is guilty and she will be correctly punished by her behaviour.
- *She is an intermediate node but she does not have the certificate for some reason.* An intermediate node should not forward a query which does not come with its related certificate. In addition to that, an intermediate node is responsible for keeping each certificate she receives.

This liability mechanism requires the users to use their own resources in keeping certificates related to past forwarded queries. This is a drawback of our proposal. We next provide a detailed analysis of the storage requirements needed by the users.

The European Directive 2006/24/EC [34] establishes that the *Telecommunications Service Providers* must store their user's communication data for a period of time between 6 months and 2 years. The authors in [35] argue that a query is formed by an average of 2.3 words and 15.5 characters. Assuming that a single Unicode character uses 2 bytes, the message to be signed will use an average of 31 bytes. If we use a signature scheme (e.g. RSA) with a key's length of 1024 bits [36], the signature length will use 128 bytes. The IP addresses of the users represent their identities in the transactions. Each IP address uses 4 bytes. Each certificate contains two identities, hence 8 bytes are used. Finally, the *time_stamp* uses 6 bytes (day, moth, year, hour, minute, second). Therefore, 971 queries represent 167.983 bytes of required storage for a single year. As explained above, these signatures must be stored for 2 years. This means that the proposed scheme requires an average of 335.966 bytes of storage in the client side. In addition to that, the certificates of each neighbour (direct connection) must be stored too. The typical length of a X.509 v3 certificate is 1 Kbyte [37]. The number of direct connections is expected to be small, hence this cost is considered negligible.

Due to the present capacity of the typical hard disk drives, we argue that these storage requirements are affordable.

The liability mechanism also adds a computational cost. The *legitimate owner* should compute the certificate (i.e. computing a digital signature). Any *intermediate node* should do the following operations: (i) verify the digital signature of the received certificate; and (ii) compute the digital signature of the new certificate. Finally, the *final node* should verify the received certificate. According to [38], the cost of computing a RSA digital signature of 1024 bits is 1.48 milliseconds (ms). The verification cost is 0.07 ms. Therefore, the liability mechanism has a cost of 1.48 ms for the *legitimate owner*, 1.55 ms for an *intermediate node* and 0.07 ms for the *final node*. If the average number of hops is ζ, there are $\zeta+1$ nodes and $\zeta-1$ *intermediate nodes*. Thus, the total cost of the liability mechanism can be computed as follows: $1.48+(\zeta-1)\cdot 1.55+0.07$ ms, i.e. $\zeta \cdot 1.55$ ms.

3.3 The proposed protocol in social networks with trust values between users

If our protocol uses a social network with *trust values* linked to the connections between users [26,28,29], the user U_i who generates q can initially decide to only ask her neighbours with a trust value greater than or equal to a certain level. In the same way, a user U_j can automatically reject any query that comes from a neighbour with a trust value below a certain level.

3.4 Preventing intermediate users from tampering with an answer

Regarding the transmission of the answer a from a user U_j who has submitted q to the WSE towards a user U_i who had initially generated q, our proposal assumes that none of the users in the path between U_j and U_i will swap the correct answer for a fake one. However, we next consider three approaches to address this situation:

- *Trustworthiness of the users.* In social networks with trust values between users [29], the *legitimate owner of the query* obtains an answer and a trust value. This value is calculated using the trust values of all the nodes which are in the path between the *final node* and the *legitimate owner*. The *legitimate owner* can give more or less trustworthiness to a certain answer depending on the trust level. Note that, the privacy of the users who are involved in this process is preserved [29].
- *Liability mechanism.* Section 3.2.1 presents a liability mechanism that guarantees the trustworthiness of the queries. We propose to use a similar liability mechanism to find the user who has modified the answer. This method does not prevent an attacker from tampering with the answer. Instead of that, it enables the *legitimate owner of the query* to detect a fake answer and trace the guilty user.

- *Methods based on mobile agents.* Mobile agents try to solve a similar problem (i.e. mobile agents must protect their route through different platforms or the results obtained in each remote platform), hence some techniques from that field might be used to address the tampering problem. This line requires further research. [39,40] are examples of the work done in this field.

4 Protocol behavior against the considered adversaries

We next explain the effect of the proposed protocol on the adversaries which have been detailed in Section 2.3:

4.1 Against the web search engine

The WSE receives all the queries from the users of the system. However, as a consequence of the proposed protocol, the WSE cannot know if a certain query has been generated by a certain user. This happens because every user submits queries that have not been generated by herself.

Let us assume that the web search engine performs a detailed analysis of all the queries received and classifies them by their content and the identity (IP address or cookies) of the senders. Also, let us assume the worst situation possible: a certain user U uses the protocol to submit the same query q several times. In this scenario, the web search engine mainly receives the same q from U and from her neighbours. It also receives q from some neighbours of the neighbours. Nevertheless, the quantity of queries that arrive from that source is likely to be insignificant in comparison with the number of queries that arrive from U and her direct neighbours. In this situation, the WSE is able to know which IP addresses form the group.

In this case, the system still preserves the privacy of U if she fulfills the privacy restrictions detailed in Section 2.4. In this way, U has a privacy level comparable to the use of k-anonymity [16], k being the number of neighbours of U.

4.2 Against a dishonest user who profiles other users

Our scheme enforces anonymity for users who are not directly related. According to that, a dishonest user U' who receives a query cannot know who was the user that generated it. U' knows neither the connections between the users (the topology of the social network) nor the number of hops from the

legitimate owner of the query to the *final node*. Each query q_i performs ζ hops on average, where $\zeta > 2$. Therefore, it is difficult to deduce the legitimate owner of the query, hence this prevents U' from generating a real profile of a certain user.

4.3 Against a selfish user

A user that systematically behaves in a selfish way will be isolated by the users who behave in a normal way and she will be forced to submit all her queries to the web search engine by her own. This has been previously explained in Section 3.1.2.

4.3.1 Sybil attack

It is worth to mention the special situation where the number of selfish users in the social network is increased by intention. This attack can only be performed by a user of the social network that generates many fake identities (Sybil attack [41]) and introduces them into the network. This large number of fake users will behave in a selfish way, trying to disrupt the functionality of the system.

This attack is not a real menace for the system because all the fake users introduced by the attacker will only have direct relationships with the attacker herself. The rest of the social network will not have direct contact with any of these fake entities. According to that, the fake users will only behave in a selfish way with the attacker who has generated them. If the attacker attempts to establish connections between the fake users and the normal users of the network, it is reasonable to assume that those normal users will reject the connection because they do not know the fake entities in the real life. As a result, this attack will not affect the social network in any way. The normal users will not notice the existence of the fake users.

4.4 Against a dishonest user who generates illegal queries

The system uses a liability mechanism (explained in Section 3.2.1) that provides partially revocable anonymity. Therefore, it prevents dishonest users from using the proposed protocol to submit illegal queries to the WSE.

Let us imagine a dishonest user U' who generates an illegal query q' and forwards it to her neighbour N_i. Also, N_i forwards q' to her own neighbour N_j. Finally, N_j submits q' to the WSE. Later, the governmental authorities

contact N_j regarding q'. In that case, N_j can use the certificate she owns to prove to the authorities that she is only an intermediate node who has forwarded q'. In this way, user anonymity vanishes, because each intermediate user publishes her direct relationship in the path until the authentic generator U' is reached. Then, U' will be properly punished for her activities.

5 Simulations

Our scheme has been simulated in different environments. Our objective was to prove the following:

- The proposed protocol succeeds in preserving the privacy of the users. In this way, the results obtained in the simulations are compared with the privacy requirements specified in Section 2.4.
- The proposed scheme is usable in practice: (i) its behaviour does not deteriorate when the number of users in the social network grows (it is scalable); and (ii) the query delay is acceptable for users who want to protect their privacy. We compare the query delay with the current proposals in the literature.

We have used different social networks in our tests. In each one, each user was connected to (i.e. was a neighbour of) a number of users ranging between 1 and 10. The precise figure was decided using the *power-law distribution*. As stated in [42], typical social networks are reasonably well approximated using this distribution.

A simulation test consisted in the following:

- Each user in the social network generates 971 queries (in 2005, each Internet user submitted, on average, 971 queries to a search engine [43]). All the queries generated by the same user have the same content. Queries from different users have different contents. As explained in Section 4, this is the worst situation possible: it is easier to profile a certain user who submits several times the same query in a scenario where no other user shares the same interest.
- All the users in the social network use the proposed protocol to submit all their queries to the web search engine.

The results of a simulation show all the queries that each user has submitted to the web search engine. These queries are classified depending on the user who initially generated them.

For each social network, 1000 tests were run and the average of these results were computed.

We have run simulations with two different kinds of users:

- *Honest users.* They collaborate with other users in submitting queries to the web search engine or forwarding them to their neighbours.
- *Selfish users.* They never submit their own queries to the web search engine, they are always forwarded to the neighbours. In addition to that, selfish users never accept queries from their neighbours.

Dishonest users (see Section 2.3) have not been considered because they follow the protocol in the same way that honest users do. The only difference between them is that dishonest users try to profile other users during that process.

Initially, we simulated a scenario where all the users were honest users. In that scenario, we tested four different social networks consisting, respectively, of 50, 100, 200 and 400 users. The two following questions were central to our proposal in these preliminary simulations:

(1) The % of exposed users (exposed users have been introduced in Section 2.4).
(2) The average number of hops that a certain query does on its travel towards the WSE. A single hop represents the process of transmitting one query from one user to another one.

The percentage of exposed users reveals whether the protocol is behaving properly (from the privacy point of view). More specifically, it is important to show that this percentage does not grow linearly with the number of users in the social network. If it grows linearly (or worse), the protocol would not be scalable, hence it would be unable to work correctly in huge networks.

The average number of hops represents a trade-off between how far a query travels in the system (several hops represents a high dissemination of a certain profile around the network, which is good from the privacy point of view) and how long it takes to submit a query to the WSE. Let δ_p represents this time interval in milliseconds (ms). Interval δ_p must be moderately close to the time interval δ_u that is needed by a single user to submit her own query to the search engine without using our scheme. We have tested that δ_u is a value around 400 ms. The fastest proposal in the literature (UUP [18]) performs this operation in 5200 ms. Schemes based on Tor perform it in 10000 ms. According to that, δ_p should be smaller than 5200 ms.

Figure 1 shows that our two initial requirements hold. In particular, the number of exposed users does not grow significantly with the number of total users in the network. Actually, in the social network of 400 users, 43.5 % of users

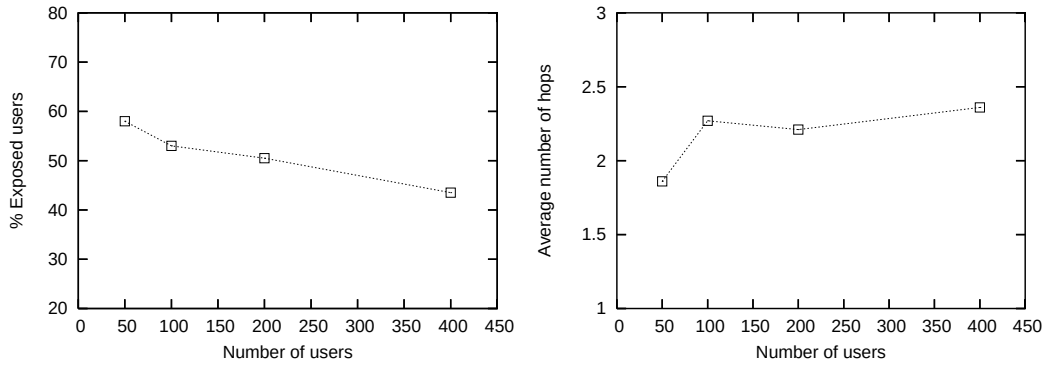

Fig. 1. Simulation scenario

are exposed. In contrast, in the network of 50 users, 58 % of users are exposed. Note that these high percentages of exposure are computed from the total number of users of the network without considering the number of direct connections that each one holds. Sections 5.1 and 5.2 extend these simulation results taking into account the connectivity level of the users. Tables in these sections prove that the users with a reasonable number of neighbours are properly protected by our scheme.

Regarding the average number of hops, results are in the time interval δ_p that we have imposed above. We have measured the communication cost between two distant nodes (one node was in Germany and the other one was in Spain) connected through the Internet. As a result, a single hop costs, on average, 732 ms (note that, generally, in a real social network, users are physically close to their *virtual* neighbours, hence this cost might be lower in a real scenario). Figure 1 shows that for 400 users, an average of 2.4 hops are needed. This is the number of hops needed to reach the user that finally submits the query to the web search engine. Thus, 4.8 hops are required to forward the answer to the user who generated the query. In addition to that, submitting the query and receiving the response requires 400 ms. Therefore, the total amount of time required without the liability mechanism is 3914 ms. If the system uses this mechanism, the total amount of time required is 3918 ms (see section 3.2.1).

This delay is significant in comparison with submitting directly a query to the WSE. However, this straightforward option does not provide anonymity to the users. In the other hand, a delay of 3914 ms is fairly below the interval of 10000 ms achieved by the approaches which are based on Tor. Note that Tor achieves 10000 ms with paths of length two (the default value is three). In addition to this large delay, the Tor approach does not consider the trade-off between the privacy level achieved by the users and the quality of service. It only takes into account the anonymity of the users. Therefore, the users may receive a deficient service. The UUP [18] achieves a delay of 5200 ms which is also worse than the one achieved by our proposal. Besides, the UUP does not take into account the quality of service either.

There are two ways to improve the delay achieved by our proposal: (i) reduce the number of hops; and (ii) reduce the communication cost between two nodes. Currently, the proposed system uses an average of 2.4 hops. This value should not be reduced because it would affect the privacy level offered to the users. Thus, the best option is to reduce the communication cost. This option depends on the technology used by the nodes. Further discussion on this topic is outside the scope of this paper.

In addition to that, the average number of hops does not grow with the number of users (it depends on the number of connections of each node and the behaviour of each neighbour). Hence, our scheme should behave properly in huge social networks.

In the following, we focus on the social network of 400 users. In the next section, we check out that the privacy requirements detailed in Section 2.4 hold when all the users are honest. Later, we analyze how the proposed protocol behaves when there is a percentage of selfish users in the social network. In this situation, only the privacy of the *honest* users must be preserved.

5.1 Simulation results from scenarios without selfish users

Table 3
Results for a social network of 400 users without selfish participants

category	% exposed users	average position	deviation
1 neighbour	$99.45\% \pm 0.12$	1.01	0.01
2 neighbours	$92.06\% \pm 0.32$	1.08	0.08
3 neighbours	$47.33\% \pm 0.39$	1.68	0.72
4 neighbours	$20.37\% \pm 0.16$	2.11	0.95
5 neighbours	$7.13\% \pm 0.24$	3.23	1.21
6 neighbours	$1.52\% \pm 0.22$	3.95	1.55
7 neighbours	$0\% \pm 0$	5.14	2.15
8 neighbours	$0\% \pm 0$	6.32	2.41
9 neighbours	$0\% \pm 0$	7.30	2.40
10 neighbours	$0\% \pm 0$	7.00	2.91

In Table 3, it can be observed that the proposed protocol provides a high level of privacy to the users with 5 or more neighbours. Users with 4 relationships are also fairly protected: only 20.37 % of these users are exposed. This table also shows that our protocol does not behave properly for users with only one or two connections (99.45 % and 92.06 % of exposed users respectively).

For each category we show the precision. The precision [44] defines an interval. The real average is within this interval with a 95% of confidence. We have calculated the precision ρ using the following expression:

$$\rho = \pm \frac{2 \cdot \mathcal{S}}{\sqrt{\mathcal{N}}}$$

where $\mathcal{S}$ is the standard deviation of the % of exposed users, and $\mathcal{N}$ is the number of simulations ($\mathcal{N} = 1000$).

The *average position* and the *deviation* of this position linked to the users with 4 or more connections hold with the privacy requirements which were introduced in Section 2.4.

As a conclusion, simulations show that the proposed scheme works fine for users with an average or a high number of direct connections. This is what one would expect: it is easier to hide inside a large group than inside a small one.

5.2 Simulation results from scenarios with selfish users

Table 4
Results on honest users for a social network of 400 users with 10 % of selfish participants

category	% exposed users	average position	deviation
1 neighbour	$98.29\% \pm 0.13$	1.03	0.03
2 neighbours	$90.91\% \pm 0.35$	1.08	0.08
3 neighbours	$48.47\% \pm 0.44$	1.68	0.29
4 neighbours	$35.80\% \pm 0.52$	2.11	1.08
5 neighbours	$9.65\% \pm 0.28$	3.23	1.27
6 neighbours	$3.12\% \pm 0.27$	3.95	1.75
7 neighbours	$0\% \pm 0$	5.14	2.13
8 neighbours	$0\% \pm 0$	6.32	2.38
9 neighbours	$0\% \pm 0$	7.30	2.10
10 neighbours	$0\% \pm 0$	7.00	2.90

In Table 4 and Table 5, it can be observed the behaviour of the protocol when 10 % of users are selfish. Table 4 shows how this situation affects the honest users. In this case, results reveal that the % of exposed users increases.

Table 5
Results on selfish users for a social network of 400 users with 10 % of selfish participants

category	% exposed users	average position	deviation
1 neighbour	100% ± 0	1.00	0.00
2 neighbours	100% ± 0	1.00	0.00
3 neighbours	100% ± 0	1.00	0.00
4 neighbours	100% ± 0	1.00	0.00
5 neighbours	100% ± 0	1.00	0.00
6 neighbours	100% ± 0	1.00	0.00
7 neighbours	100% ± 0	1.00	0.00
8 neighbours	100% ± 0	1.00	0.00
9 neighbours	100% ± 0	1.00	0.00
10 neighbours	100% ± 0	1.00	0.00

Nevertheless, all the users with more than 6 connections keep their privacy perfectly. Users with 5 and 6 connections are also properly protected. Note that the users with 4 neighbours are in a worse situation now, but they are still fairly protected (35.80% of exposed users).

Regarding the selfish users, Table 5 shows that all the selfish users present in the scenario are profiled by the WSE (their privacy is not preserved). According to that, our scheme behaves properly: all selfish users are punished. This incentivizes users to follow the proposed protocol correctly.

Table 6, Table 7 and Table 8 show the behaviour of the protocol when 20 %, 40 % and 80 % of users are selfish, respectively. It can be observed in these results that when the number of selfish users increases, the % of exposed users (honest users) increases as well. Users with several neighbours are not safe from this situation.

More specifically, in Table 8 privacy is only guaranteed to the users with 10 connections. The reason is that, due to the large number of selfish users, honest users may be surrounded by selfish ones. As a result, those users will become isolated and they will get low levels of protection.

The results shown by Table 8 depend on the number of users in each category of connections (e.g users with 2 neighbours, users with 3 neighbours...) and the number of selfish neighbours that each honest user has (level of isolation). Selfish users were selected at random and for this reason it is possible to have a honest user with 6 neighbours who is less isolated than a honest user with

Table 6
Results on honest users for a social network of 400 users with 20 % of selfish participants

category	% exposed users	average position	deviation
1 neighbour	$99.57\% \pm 0.14$	1.02	0.02
2 neighbours	$87.33\% \pm 0.37$	1.15	0.14
3 neighbours	$55.10\% \pm 0.47$	1.61	0.40
4 neighbours	$36.02\% \pm 0.51$	2.07	1.09
5 neighbours	$15.35\% \pm 0.54$	2.76	1.47
6 neighbours	$4.33\% \pm 0.31$	3.64	1.99
7 neighbours	$1.84\% \pm 0.37$	4.19	2.60
8 neighbours	$0\% \pm 0$	5.36	2.54
9 neighbours	$0\% \pm 0$	5.57	1.87
10 neighbours	$0\% \pm 0$	6.50	2.02

Table 7
Results on honest users for a social network of 400 users with 40 % of selfish participants

category	% exposed users	average position	deviation
1 neighbour	$97.43\% \pm 0$	1.02	0.02
2 neighbours	$82.37\% \pm 0.67$	1.15	0.13
3 neighbours	$62.57\% \pm 0.52$	1.61	0.21
4 neighbours	$33.82\% \pm 0.58$	2.07	0.80
5 neighbours	$32.68\% \pm 0.33$	2.76	0.82
6 neighbours	$9.21\% \pm 0.32$	3.64	1.07
7 neighbours	$5.36\% \pm 0.14$	4.19	1.93
8 neighbours	$0\% \pm 0$	5.36	2.42
9 neighbours	$0\% \pm 0$	5.57	2.51
10 neighbours	$0\% \pm 0$	6.50	2.88

7 neighbours. Note that there is a really low number of honest users in each category and this fact cause the results to vary a lot between categories (e.g only 40% of the users with 6 connections were exposed but 75% of the users with 7 connections were exposed).

As a conclusion, Table 8 shows that when the percentage of selfish users is high (social network of 400 users with 80 % of selfish participants), our proposal

Table 8
Results on honest users for a social network of 400 users with 80 % of selfish participants

category	% exposed users	average position	deviation
1 neighbour	$100\% \pm 0$	1.00	0.00
2 neighbours	$60.42\% \pm 1.17$	1.36	0.34
3 neighbours	$74.75\% \pm 0.27$	1.25	0.22
4 neighbours	$58.83\% \pm 0.37$	1.50	0.47
5 neighbours	$50\% \pm 0$	1.80	0.30
6 neighbours	$40\% \pm 0.66$	1.60	0.58
7 neighbours	$75\% \pm 0$	1.25	0.60
8 neighbours	$40\% \pm 0$	1.80	0.37
9 neighbours	$17.66\% \pm 0.75$	2.33	0.41
10 neighbours	$0\% \pm 0$	2.00	0.40

does not protect properly the privacy of the users that follow the protocol. In this situation, only the honest users with 10 connections have enough chances to stay safe from isolation. Nevertheless, we claim that the % of selfish users is expected to be low because all selfish users are unable to protect their privacy (they are punished by the system). According to that, we consider that these results are acceptable.

5.3 Comparison with a random system

In this section, we compare the presented system with a scheme that follows the same protocol but it randomly chooses which queries should be submitted/forwarded and to whom they should be sent. That is, if a certain user N_i has k neighbours, she sends her own queries to the WSE with a probability of $1/(k+1)$, and forwards them to one of her k neighbours with the same probability.

Both schemes use the same mechanism to prevent selfish users from damaging the system. The purpose of this comparison is to show how our procedure overcomes a straightforward proposal that follows a similar approach.

The random system was simulated with the same set-up as our proposal in a social network with 400 honest users. As a result, all the users were exposed. Even the users with 10 connections were profiled by the WSE. This situation happens because the users do not consider the profile exposure level. Instead

of that, they distribute uniformly their queries to their neighbours and these neighbours also distribute them to their own connections. The result of this process is that these users submit significantly less queries to the WSE than the origin user. Thus, the origin user's profile is not hidden among her group of neighbours.

In addition to that, the number of queries sent to each user is not considered either. Therefore, users with only a few connections can send too many queries to the same neighbours. This may activate the mechanism that prevents selfishness. If it happens, these users will be exposed.

We next compare the profile of a certain user U with 5 connections when she uses our proposal and when she uses the random scheme. Table 9 shows the proportion of queries submitted by U which have been generated by U herself (Own), by her neighbours and by the rest of the social network. This table presents the differences between the two systems. It can be observed that in our proposal, the proportion of queries generated and submitted by U (Own) was hidden among her group of neighbours. In the random approach, U was exposed.

Table 9
Proportion of queries submitted by a certain user with both systems.

	Random proposal	Our proposal
Own	33.19 %	6.29 %
1st neighbour	8.88 %	25.10 %
2nd neighbour	4.88 %	8.67 %
3rd neighbour	5.72 %	12.80 %
4th neighbour	7.76 %	6.07%
5th neighbour	3.75 %	3.84 %
Others	35.82 %	37.23 %

6 Deployability issues

Our approach uses a special type of social network where anonymity is enforced for nodes which are not directly related. This particular type of social networks was introduced in [29].

In the same way, users of *instant messaging* applications are only aware of their own neighbours. Examples of this kind of applications are *Windows Live Messenger*, *Skype* or *Google Talk* among others.

According to that, the *ideal* approach would be to integrate our scheme into one of these already developed instant messaging applications. However, these well-known infrastructures depend on a central entity which knows (or is able to know) all relations between users. If that central entity is honest and preserves the privacy of the users (it does not share any information about the users with web search engines or other entities), this solution is the best for implementing our protocol.

Nevertheless, we consider that the trustworthiness of the central entity usually cannot be guaranteed. Note that it is common for the central entity and the web search engine to be owned by the same party (for example *Google talk* and *Google*). In that case, we require a distributed social network (without a central node that manages it) which supports our scheme.

The social networks that we envisage are formed by several nodes which are connected between them. These nodes are expected to be permanently connected to the network due to the use of flat-rate broadband connections to the Internet. Nevertheless, some of these nodes can become off-line at some point for several reasons. Even worse, these nodes can become on-line again with a different IP address. That situation can affect the connectivity between nodes and, as a result, the social network functionality.

We next explain the behaviour of our distributed social network (DSN) in three common situations:

- *A user is invited to join the DSN.* There is no central node, thus a user U_i, willing to join a DSN for first time, should be invited by a current member U_j of the DSN. According to that, U_i should have some kind of relation in real life with U_j prior to become part of the DSN. Then, that real relation is represented in the DSN by a direct connection between both users. To establish a direct connection, two users must exchange their current IP addresses. It can be easily done by sending an invitation e-mail from U_j to U_i. To guarantee security and authentication between the two parties, we propose to use PGP [45] or a similar application. At the end of this process, each user writes down the IP address of the other in her neighbour address list.
- *A user interacts with the DSN.* Let us imagine a user U_i who wants to forward a query q to one of her neighbours. In this situation, U_i will select one IP address from her neighbour address list. Then, U_i will use that IP address to contact the selected user and ask her to accept q.
- *A user becomes on-line in the DSN with a different IP address.* There is no central node managing the DSN and users rely on their neighbour address list to get in touch. This means that a certain user who gets her IP address changed for some reason must tell her neighbours (she can do it by using her neighbour address list) about this modification. There is a problem when

two users (U_i and U_j) directly connected change their addresses at the same time. In this case, U_i must tell U_j about the modification but U_j's IP address is outdated too. The same occurs in U_j's side. The straightforward solution to re-establish the connection between U_i and U_j is to send the invitation via e-mail like when a user in invited to join the DSN for the first time.

- *A participating user goes off-line in the middle of a protocol's execution.* Our proposal works without direct interaction from the users. Thus, they can disconnect their computers while the system is providing service to others. As a result, some participating users might go off-line in the middle of a query/answer forwarding. This situation can be harmful for the system.

  The straightforward solution to this issue is the use of time-outs like the ones used in the *TCP/IP protocol*. In this way, if an answer has not been received before a certain timeout, the user who has generated the query sends it again. This process must be executed by the origin user because she is the one interested in receiving the answer and she cannot know which user has become off-line.

  Other possible solutions are the following: (i) send the same query to several users. This solution can create important problems (by flooding) to the network, hence it should be carefully evaluated before being deployed; and (ii) in a social network with trust values, they can be used to punish the unreliable users. Note that, users who occasionally interrupt some queries can recover their trust values if they handle new requests from other users.

7 Concluding remarks

Protecting the privacy of the users who use web search engines is an important issue. Nevertheless, this protection should be offered without harming the service provided to the users. This problem has not been deeply addressed in the literature.

We have described in this paper a new protocol which preserves the privacy of the users by distorting their profiles. The distorted profiles still allow the users to get a proper service from the WSEs.

Our proposal is based on the two following assumptions: (i) the proliferation of home computers equipped with flat-rate broadband connection to the Internet. This implies that computers can be permanently connected to the network; and (ii) the gradual introduction of social networks.

The simulated performance of our proposal shows that it can be successfully deployed in real environments because: (i) it provides an acceptable query delay; (ii) it offers privacy to users who have a reasonable number of direct connections; and (iii) it works properly in scenarios with users who do not

follow the protocol.

Disclaimer and acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments that helped to improve the quality of this paper. Also, they thank Josep Maria Mateo-Sanz for his help with the simulation analysis.

The authors are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. This work was partly supported by the Spanish Ministry of Education through projects TSI2007-65406-C03-01 "E-AEGIS" and CONSOLIDER CSD2007-00004 "ARES", by the Spanish Ministry of Industry, Commerce and Tourism through project TSI-020100-2009-720 and by the Government of Catalonia under grant 2009 SGR 1135.

References

[1] iProspect.com, Inc,. "iProspect Blended Search Results Study", 2008. `http://www.iprospect.com`

[2] K. Sugiyama, K. Hatano, M. Yoshikawa, "Adaptive Web Search based on user profile constructed without any effort from users", *Proc. of the 13th International World Wide Web Conference*, 2004.

[3] F. Qiu, J. Cho, "Automatic identification of user interest for personalized search", *Proc. of the 12th International World Wide Web Conference*, 2006.

[4] M. Speretta, S. Gauch, "Personalizing search based on user search history", *Proc. of International Conference of Knowledge Management –CIKM'04*, 2004.

[5] J. Teevan, S. T. Dumais, E. Horvitz, "Personalizing search via automated analysis of interests and activities", *Proc. of 28th Annual International ACM Conference on Research and Development in Information Retrieval –SIGIR'05*, 2005.

[6] Google personalized search, 2009. `http://www.google.com/psearch`

[7] M. Barbaro, T. Zeller. "A face is exposed for AOL searcher No 4417749", *New York Times*, August 2006.

[8] J. Pitkow, H. Schuetze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, T. Breuel, "Personalized search", *Communications of the ACM*, vol. 45, no. 9, pp. 50–55, 2002.

[9] G. Jeh, J. Widom, "Scaling personalized web search", *Proc. of the 12th International World Wide Web Conference*, 2003.

[10] F. Saint-Jean, A. Johnson, D. Boneh, J. Feigenbaum, "Private Web Search", *Proc. of the ACM workshop on Privacy in electronic society – WPES'07*, pp. 84–90, 2007.

[11] X. Shen, B. Tan, C.X. Zhai, "Privacy Protection in Personalized Search", *ACM SIGIR Forum*, vol. 41, no. 1, pp. 4–17, 2007.

[12] Y. Xu, B. Zhang, Z. Chen, K. Wang, "Privacy-Enhancing Personalized Web Search", *Proc. of the 16th international conference on World Wide Web*, pp. 591–600, 2007.

[13] TrackMeNot, 2009. `http://mrl.nyu.edu/ dhowe/trackmenot`

[14] G. T. Marx, "A Tack in the Shoe: Neutralizing and Resisting the New Surveillance" *Journal of Social Issues*, vol. 59, no. 2, pp. 369–390, 2003.

[15] J. Domingo-Ferrer, A. Solanas, J. Castellà-Roca, "$h(k)$-Private Information Retrieval from Privacy-Uncooperative Queryable Databases", *Journal of Online Information Review*, vol. 33, no. 4, pp. 1468-4527, 2009.

[16] L. Sweeney, "k-anonymity: a model for protecting privacy", *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, pp. 557-570, 2002.

[17] J. Domingo-Ferrer, M. Bras-Amorós, Q. Wu, J. Manjón, "User-Private Information Retrieval Based on a Peer-to-Peer Community", *Data and Knowledge Engineering*, to appear.

[18] J. Castellà-Roca, A. Viejo, J. Herrera-Joancomartí, "Preserving user's privacy in web search engines", *Computer Communications*, vol. 32, no. 13–14, pp. 1541–1551, 2009.

[19] The Tor Project, 2009. `http://www.torproject.org`

[20] Foxtor, 2009. `http://cups.cs.cmu.edu/foxtor`

[21] Torbutton, 2009. `http://freehaven.net/~squires/torbutton`

[22] B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan, "Private information retrieval", *IEEE Symposium on Foundations of Computer Science – FOCS*, pages 41-50, 1995.

[23] B. Chor, O. Goldreich, E. Kushilevitz, M. Sudan, "Private information retrieval", *Journal of the ACM*, vol. 45, pp. 965–981, 1998.

[24] E. Kushilevitz, R. Ostrovsky, "Replication is not needed: single database, computationally-private information retrieval", *Proc. of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pp. 364–373, 1997.

[25] R. Ostrovsky, W. E. Skeith-III, "A survey of single-database pir: techniques and applications", *Lecture Notes in Computer Science*, vol. 4450, pp. 393–411, 2007.

[26] J. Domingo-Ferrer, "A public-key protocol for social networks with private relationships", *Lecture Notes in Computer Science*, vol. 4617, pp. 373–379, 2007.

[27] The official Google blog, `http://googleblog.blogspot.com`

[28] B. Carminati, E. Ferrari, A. Perego, "Private relationships in social networks", *Private Data Management – PDM'07*, 2007.

[29] J. Domingo-Ferrer, A. Viejo, F. Sebé, U. González-Nicolás, "Privacy homomorphisms for social networks with private relationships", *Computer Networks*, vol. 52, no. 15, pp. 3007–3016, 2008.

[30] Open Directory Project, 2009. `http://www.dmoz.org/`

[31] D. Brewer, S. Thirumalai, K. Gomadamk, K. Li, "Towards an Ontology Driven Spam Filter", *Proceedings of the 22nd International Conference on Data Engineering Workshops*, 2006.

[32] S. Youn, D. McLeod, "Efficient Spam Email Filtering using Adaptive Ontology", *Proceedings of the International Conference on Information Technology*, pp. 249–254, 2007.

[33] European Comission, "The Legal and Market Aspects of Electronic Signatures", 2009. `http://ec.europa.eu`

[34] European Parliament and Council, "Directive on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks", 2006. `http://eur-lex.europa.eu/`

[35] M. Kamvar and S. Baluja, "A large scale study of wireless search behavior: Google mobile search", *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 701–709, 2006.

[36] Recommendation for Key Management, Special Publication 800–57 Part 1, NIST, 2007.

[37] R. Housley, W. Polk, W. Ford, D. Solo, "Internet X.509 Public Key Infrastructure Certificate", Internet RFC 3280, 2002. `http://www.ietf.org`

[38] Crypto++ 5.6.0 Benchmarks, 2009.
`http://www.cryptopp.com/benchmarks.html`.

[39] P. Maggi, R. Sisto, "A configurable mobile agent data protection protocol", *Proc. of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 851–858, 2003.

[40] J. Zhou, J. A. Onieva, J. Lopez, "Analysis of a free roaming agent result-truncation defense scheme" *Proc. of the IEEE Int. Conf. on e-Commerce Technology*, pp. 221–226, 2004.

[41] J.R. Douceur, "The Sybil Attack", *Lecture Notes in Computer Science*, vol. 2429, pp. 251–260, 2002.

[42] D. Liben-Nowell, "An algorithmic approach to social networks", Ph.D. Thesis, MIT Computer Science and Artificial Intelligence Laboratory, 2005.

[43] Internet World Stats, 2008. `http://www.internetworldstats.com`

[44] G. C. Canavos, "Applied Probability and Statistical Methods", Little Brown & Co, ISBN 0673390195, 1984.

[45] S. Garfinkel, "PGP: Pretty Good Privacy", *O'Reilly & Associates*, 1995.