# Rational Behavior in Peer-to-Peer Profile Obfuscation for Anonymous Keyword Search: the Multi-Hop Scenario

Josep Domingo-Ferrer and Úrsula González-Nicolás

*Universitat Rovira i Virgili*
*Department of Computer Engineering and Mathematics*
*UNESCO Chair in Data Privacy*
*Av. Països Catalans 26, E-43007 Tarragona, Catalonia*
*Tel.: +34 977558270*
*Fax: +34 977559710*
*E-mail {josep.domingo,ursula.gonzaleznicolas}@urv.cat*

**Abstract**

Web search engines (WSEs) have become an essential tool for searching the huge amount of information stored in the World Wide Web. WSEs try to build query profiles of their users, in order to increase the accuracy of the results provided to users and also to fine-tune advertising. Profiling the query interests of users clearly encroaches on their privacy. There are several anti-profiling approaches in web search, among which those relying on peer-to-peer profile obfuscation stand out. In this paper, we analyze the multi-hop peer-to-peer profile obfuscation game, in which a peer forwards her query to a second peer, who may submit it on behalf of the first peer or just forward it to a third peer, and so on. We describe several privacy utility functions for peers and a rational protocol in terms of a generic privacy utility function. It turns out that rational behavior leads peers to helping each other.

*Key words:*
Anti-profiling; Private information retrieval; Anonymous keyword search; User-private information retrieval; Multi-hop peer-to-peer profile obfuscation; Game theory.

## 1 Introduction

Web search engines (WSEs) are essential to manage the huge amount of information available on the Internet, scattered over hundreds of millions of web

sites. Since the majority of WSE users tend to click only on the first results in the first search result page, it is important for WSEs to find out what are the interests of users. For example, for a user with unknown interests, a search for "Barcelona" should return links on the city at the top, while for a user interested in soccer, links at the top should correspond to the soccer club, and for a user interested in architecture, links at the top should be about famous buildings in the city. Every user has her own interests, and these may change over time.

Since users do not usually collaborate with WSEs by explicitly defining what their interests are, WSEs may try to infer those interests using the browsing history. The cost of such profiling by the WSE is more than compensated by the possibility to embed in the search results advertising tailored to the user's interests. The problem with user profiling is that the query history of a user may contain private information: *e.g.* someone frequently looking for a disease or a drug leaks information about her likely health condition. On the other hand, even if users of WSEs do not identify themselves, it is not difficult to establish the identity of the individual to whom a query history corresponds, as it became evident in the AOL search data scandal [2].

### *1.1 Contribution and plan of this paper*

As argued in Section 2 below, peer-to-peer (P2P) query profile obfuscation protocols stand out as a competitive anti-profiling approach in web search. In such protocols, a user obfuscates her profile with the help of a P2P community: the user submits queries on behalf of her anonymous peers and conversely. There are two variants of P2P profile obfuscation systems: *single-hop* and *multi-hop*. In a single-hop system, when a peer requests help from a second peer in submitting a query to the WSE (in order to hide the first peer's interests from the WSE), the second peer either grants that help and submits the query or she refuses to submit it. In a multi-hop system, the second peer either submits the query as requested or she forwards the query to a third peer, who may submit it or forward it further, and so on.

The main potential shortcoming is why should peers be interested in helping other peers to preserve their query privacy. This paper addresses that issue by characterizing what is the rational behavior of peers in a *multi-hop* P2P profile obfuscation system (the single-hop case has recently been dealt with in [11]).

Regardless of whether the peers are a community of acquaintances or not, we assume that, when a peer forwards a query to another peer, the forwarder and the receiver do not know each other's identities. This assumption is reason-

able because the receiver has no idea whether the received query was originated by the forwarder or was originated by some other peer and is merely forwarded by the forwarder. Hence, re-identification by profile is thwarted. If re-identification by IP address among peers is a concern, a system like Tor [30] might be used to provide transport-level anonymity in peer-to-peer communication.

The contributions in this article are as follows. We give a game-theoretic analysis of *multi-hop* P2P profile obfuscation systems like Crowds ([27]), TorTorbutton ([31]) or the one in [32]. We specify a metric for the privacy of peers vs the WSE. The main difference with the analysis of single-hop P2P systems contributed in [11] is that in a multi-hop setting the privacy of peers vs other peers is implicit. We compute the privacy utilities of the different strategies that can be followed by the peers in view of maximizing their privacy. Maximizing the privacy utility when deciding what to do with each new query yields a rational behavior for each peer, which allows automating the decision process at the peers. In particular, the conditions under which a Nash equilibrium [21,22] among peers exists are determined; under such an equilibrium, the best option for one peer to maximize her privacy is to have her query submitted by another peer, and the best option for the other peer to maximize his privacy is to submit the first peer's query to the WSE.

Section 2 describes related work on anti-profiling in database or web search, including P2P profile obfuscation. Section 3 discusses the pros and cons of multi-hop vs single-hop P2P profile obfuscation. Section 4 gives some background on game theory. Section 5 proposes a characterization of the multi-hop P2P profile obfuscation game and proposes several possible privacy utility functions. Simulation results are reported in Section 6. Section 7 lists the conclusions of this paper.

## 2 Related work

A user can fight profiling in database or web search by using a variety of approaches, including: i) basic techniques like dynamic IP addresses and/or cookie rejection; ii) querying under pseudonym; iii) standalone profile obfuscation [14,20,17]; iv) proxy-based obfuscation [16]; v) P2P profile obfuscation, either single-hop [10,4,29] or multi-hop, *e.g.* based on Tor with the Torbutton add-on [30,31], or Crowds [27] or [32]); vi) private information retrieval [6,7,25,1].

Let us briefly analyze the shortcomings of the above approaches (see [11] for a more detailed analysis):

- Basic techniques are often problematic: *e.g.* the user is unlikely to control the renewal policy for dynamic IP addresses, and rejecting cookies may entail an unacceptable loss of functionality.
- Submitting queries using pseudonyms offers only weak protection, as the search engine can profile pseudonyms and eventually link them to the real user's identity (as illustrated by the re-identification success from the pseudonymized AOL query logs [2]).
- Standalone profile obfuscation requires either supplementing queries with fake keywords or generating entire additional fake queries to cloak the real ones; on the one hand, supplementing with fake keywords is far from obvious, as these must be semantically similar enough to the real keywords for the latter to be indistinguishable; on the other hand, generating entire additional fake queries overloads search engines and communications.
- Proxy-based profile obfuscation is predicated on submitting queries to a search engine through a proxy, who strips them of any identifying information before forwarding them to the search engine. The proxy may even accept to forward encrypted queries to the search engines, in which case the proxy does not see the content of the queries. The weakness of this approach is that a collusion between the proxy and the search engine is enough to compromise the profiles of all proxy users.
- P2P profile obfuscation has the shortcoming of requiring the co-operation of peers and, depending on the implementation, it may also entail some privacy loss for the user vs her peers.
- The objective of private information retrieval is to enable a user to retrieve an information item from a database without the latter learning which item the user was interested in. Hence, it targets the most ambitious privacy level. PIR protocols proposed so far have some fundamental shortcomings which hinder their practical deployment:

(1) If the database contains $n$ items, theoretical PIR protocols have $O(n)$ complexity [6,7]: the protocol must "touch" all records to avoid giving the server any clues on the value of $i$; this is unaffordable for large databases and/or search engines [3].
(2) It is assumed that the database server cooperates in the PIR protocol; it is the user who is interested in her own privacy, whereas the motivation for the database server is dubious, not to say contrary to user privacy.
(3) In general and especially in the case of search engines, the database cannot be modeled as a vector in which the user knows the physical address $i$ of the item sought; even keyword PIR [5,24] does not really fit, as it still assumes a mapping between individual keywords and physical addresses.

In view of the above, P2P profile obfuscation can be a good option if we are able to ensure that peers are interested in co-operating and at the same time we can minimize the privacy loss of the user versus her peers. This is precisely the purpose of this paper, which focuses on rational behavior in multi-hop systems (see [11] for an analysis of the single-hop case).

## 3 Multi-hop vs single-hop P2P profile obfuscation

Before going any further, let us compare the pros and cons of single-hop and multi-hop obfuscation. In a multi-hop system, a query may pass through several intermediate peers before being submitted, hence if two peers $A$ and $B$ exchange a query, they cannot reconstruct and/or anticipate the rational decisions by each other, because those decisions may involve third peers. This has advantages and disadvantages.

A problem of single-hop is that any peer $B$ receiving a query from another peer $A$ knows that $A$ was the query originator. In contrast, multi-hop offers increased privacy of the user's query profile vs her peers: a peer $B$ who receives a query from a peer $A$ does not know whether the query was originated or merely forwarded by $A$, and that gives peer $A$ privacy vs peer $B$. Let us examine the chances of a collusion of peers to determine the query originator in the multi-hop scenario. Since no single peer other than the query originator knows who originated a query, for a collusion to succeed in determining the query originator, the collusion size should be $N-1$: if the $N-1$ colluders *loyally* tell each other that they did not originate a certain query, then all of them know that the query was originated by the only non-colluding peer. However, collusions of size $N-1$ are unlikely for moderate to large $N$: the larger $N$, the more difficult it is to organize a collusion of $N-1$ peers and the easier it is for the non-colluding peer to learn about the collusion.

A weakness of letting a query (and its response) travel several hops is that it is exposed to more peers. Indeed, imagine that the query is about patents on a certain type of device. Certainly, none of the peers who receive the query knows who originated it, but they all know that there is someone currently interested on patents for that device type. This constitutes a leakage of industrial strategy, especially if there are several successive refinements of queries on that device type. At a first glance, single-hop would seem less disclosive than multi-hop in this respect. However, single-hop may require several attempts by the query originator before a peer willing to submit the query is found. Hence, in single-hop, the query may also be exposed to several peers, even there is the small advantage that the query originator selects and knows who those peers are.

Another downside of allowing multiple hops are longer response times. If $A$ originates and forwards a query to $B$, $A$ cannot predict whether $B$'s rational decision will be to submit the query or to forward it. Hence, the average time for the query originator to obtain a response can be quite long. Again, at first glance single-hop seems to outperform multi-hop in this respect. However, since single-hop may also need several attempts before a collaborative peer is found, the response time can also be quite long. The advantage of single-hop is

that the query originator can decide how many peers she tries before submitting herself her own query. For multi-hop, some kind of timeout or maximum number of hops may also be imposed in view of bounding the response time (see Algorithm 1 below in this paper).

The fact that the number of attempts in single-hop and the number of hops in multi-hop are unpredictable *ex ante* represents a common privacy advantage of both approaches vs the database/WSE: it is more difficult for the latter to link the queries from the same originator based on their timing, because the delay between query generation and query submission is variable.

A final problem with single-hop is that, if no helping peer is found in a reasonable number of attempts, the user having originated the query will have to submit it herself, even if doing so reduces the obfuscation level of her actual query interests: *e.g.* if a user submits a query many more times than other queries, the adversary infers that this query must really have been originated by that user, because the user would not submit that query on behalf of another user (if we assume that all users wish to exhibit a "flat" profile of submitted queries to hide their actual interests). In contrast, in multi-hop with a timeout or a maximum number of hops, the user originating a query and forwarding it can be assured that someone will submit her query; there is of course a chance that, after some hops, the originating user receives back her own forwarded query, which she decides to submit, but in this case, the database/WSE cannot distinguish whether the submitting user was the originating one.

One could argue that the above problem of single-hop could be fixed by forcing peers receiving a help request to accept it via timeout or maximum number of attempts. Note however that this would be more unfair than in multi-hop: whereas in multi-hop the originating peer cannot know who (if anyone) will be forced by timeout to submit her forwarded query, in single-hop the originating peer *decides* which peer she asks when the timeout or the maximum number of attempts are about to be reached, so an originating peer could deliberately disrupt the profile of another peer if the latter could not reject a help request.

From the above analysis, both single-hop and multi-hop have relative strengths and weaknesses, but, overall, multi-hop seems to have more strong points.

## 4 Basics of game theory

A game is a protocol between a set of $N$ *players*, $\{1, \cdots, N\}$. Each player $i$ has her own *set of possible strategies*, say $S_i$. To play the game, each player $i$ selects a strategy $s_i \in S_i$. We will use $s = (s_1, \cdots, s_N)$ to denote the vector of

strategies selected by the players and $S = \Pi_i S_i$ to denote the set of all possible ways in which players can pick strategies.

The vector of strategies $s \in S$ selected by the players determines the outcome for each player, which can be a payoff or a cost. In general, the outcome will be different for different players. To specify the game, we need to state for each player a preference ordering on these outcomes by giving a complete, transitive, reflexive binary relation on the set of all strategy vectors $S$. The simplest way to assign preferences is by assigning, for each player, a value for each outcome representing the payoff of the outcome (a negative payoff can be used to represent a cost). A function whereby player $i$ assigns a payoff to each outcome is called a utility function and is denoted by $u_i : S \longrightarrow \mathbb{R}$.

For a strategy vector $s \in S$, we use $s_i$ to denote the strategy played by player $i$ and $s_{-i}$ to denote the $(n-1)$-dimensional vector of the strategies played by all other players. With this notation, the utility $u_i(s)$ can also be expressed as $u_i(s_i, s_{-i})$.

A strategy vector $s \in S$ is a *dominant strategy solution* if, for each player $i$ and each alternate strategy vector $s' \in S$, it holds that

$$u_i(s_i, s'_{-i}) \geq u_i(s'_i, s'_{-i}) \tag{1}$$

In plain words, a dominant strategy $s$ is the best strategy for each player $i$, independently of the strategies played by all other players.

A strategy vector $s \in S$ is said to be a *Nash equilibrium* if, for all players $i$ and each alternate strategy $s'_i \in S_i$, it holds that

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$$

In plain words, no player $i$ can change her chosen strategy from $s_i$ to $s'_i$ and thereby improve her payoff, assuming that all other players stick to the strategies they have chosen in $s$. A Nash equilibrium is self-enforcing in the sense that once the players are playing such a solution, it is in every player's best interest to stick to her strategy. Clearly, a dominant strategy solution is a Nash equilibrium. Moreover, if the solution is strictly dominant (*i.e.* when the inequality in Expression (1) is strict), it is also the unique Nash equilibrium. See [22] for further background on game theory.

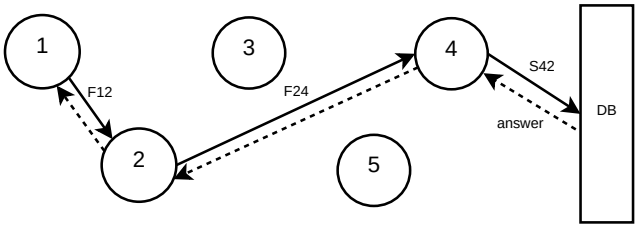


Fig. 1. Example of multi-hop strategy in a game with 5 players

## 5 The multi-hop P2P profile obfuscation game

Consider a system with $N$ players $Q^1, \cdots, Q^N$, and a WSE or database server $DB$. For any $i$, assume that $Q^i$ originates a query. Then $Q^i$ has two possible strategies:

$Sii$**:** $Q^i$ submits her query directly to $DB$;
$Fij$**:** $Q^i$ forwards her query to $Q^j$, for some $j \neq i$, and requests $Q^j$ to submit the query or have it submitted by some other player.

When receiving $Q^i$'s query, $Q^j$ has two possible strategies:

$Sji$**:** $Q^j$ submits the query received from $Q^i$ to $DB$ and returns the answer to $Q^i$;
$Fjk$**:** $Q^j$ forwards the query received from $Q^i$ to $Q^k$, for some $k \notin \{i, j\}$, and requests $Q^k$ to submit the query or have it submitted by some other player.

The above set of strategies result in a multi-hop mechanism where an originated query is forwarded from one player to another player until it is submitted to $DB$, as illustrated in Figure 1. Then, the answer is returned following the reverse path. We assume that all players along the path starting at the query originator and ending at the query submitter are *honest-but-curious* in the sense that:

- Any player $Q^j$ receiving a query from any other player $Q^i$ either submits it to $DB$ or forwards it to some other player $Q^k$;
- If a player $Q^j$ chooses to submit to $DB$ a query she did not originate, then the player returns the correct answer obtained from $DB$ to the player $Q^i$ the query came from;
- If a player $Q^j$ chooses to forward to another player $Q^k$ a query $Q^j$ did not originate, then, as soon as $Q^j$ receives the query answer from $Q^k$, $Q^j$ returns the query answer back to the player $Q^i$ the query came from;
- $Q^j$ may read and store all received queries and answers.

In this scheme, when a player $Q^j$ receives a query from $Q^i$, she does *not* know whether $Q^i$ is the query originator or just an intermediate forwarding player. This is a *major* advantage with respect to the single-hop scenario, in which,

when $Q^j$ receives a query from $Q^i$, $Q^j$ knows that $Q^i$ is the originator and therefore can use the query to profile $Q^i$'s interests. For that reason, in the single-hop scenario, it is important for $Q^i$ to distribute the forwarded queries as evenly as possible among the rest of players, in order to reduce the profiling abilities of players (see [11]). In the multi-hop scenario, the query originator $Q^i$ can randomly select the player $Q^j$ to whom she forwards her query, because, for the above reasons, privacy vs any particular player is no longer an issue.

### *5.1 The privacy model*

Let $X^i(t) = \{x^i_1, \cdots, x^i_{m_i(t)}\}$ be the set of queries originated by $Q^i$ up to time $t$. Let $Y^i(t) = \{y^i_1, \cdots, y^i_{n_i(t)}\}$ be the set of queries submitted by $Q^i$ to $DB$ up to time $t$. In both $X^i(t)$ and $Y^i(t)$, repeated queries appear repeatedly.

The privacy utility function for $Q^i$ should reflect the following intuition: the more diverse the queries in $Y^i(t)$, the more private stay the interests of $Q^i$ vs $DB$. In this approach, the interests of $Q^i$ vs any specific peer are already private, because peers do not know who is the originator of a query. Let $U_i(Y^i(t))$ be the privacy utility function for $Q^i$. For the sake of simplicity, we will assume that all players $Q^i$ choose the same privacy utility function, *i.e.* $U_1(\cdot) = \cdots = U_N(\cdot) = U(\cdot)$.

The above utility function $U(\cdot)$ evaluated on $Y^i(t)$ induces a utility function $u_i(\cdot)$ on the strategies available to $Q^i$ (as discussed in Section 4, the purpose of the utility $u_i(\cdot)$ is to allow $Q^i$ to choose the best strategy). When $Q^i$ originates a query $x^i_r$ at time $t+1$:

- $Q^i$ chooses $Sii$ (direct submission) if $U(Y^i(t+1)) \geq U(Y^i(t))$, where $Y^i(t+1) = Y^i(t) \cup \{x^i_r\}$;
- Otherwise $Q^i$ chooses $Fij$ (forwarding the query to $Q^j$), where $j \neq i$ is randomly chosen among the available peers.

When $Q^j$ receives $x^i_r$, it proceeds as follows:

- $Q^j$ chooses $Sji$ (submitting $x^i_r$ and returning the answer to $Q^i$) if $U(Y^j(t+1)) > U(Y^j(t))$, where $Y^j(t+1) = Y^j(t) \cup \{x^i_r\}$;
- Otherwise $Q^j$ chooses $Fjk$ (forwarding the query to $Q^k$), where $k \notin \{i, j\}$ is randomly chosen among the available peers.

In plain words, a peer submits the query $x^i_r$ to $DB$ if doing so increases her privacy utility vs $DB$. Otherwise, the query is forwarded to another (randomly chosen) peer.

### *5.2 Some possible privacy utility functions*

#### *5.2.1 Shannon's entropy at the query level*

Given a random variable $Z$ taking values $z_1, z_2, \cdots, z_n$ with probabilities $p_1, p_2, \cdots, p_n$, respectively, Shannon's entropy [28] is a measure of uncertainty defined as

$$H(Z) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

The more homogeneous the $p_i$, the higher is $H(Z)$: the rationale is that the outcome of $Z$ becomes more uncertain as the $p_i$ become more homogeneous. The maximum $H(Z)$ is reached when $p_1 = \cdots = p_n = 1/n$.

We have mentioned above that, the more diverse the queries in $Y^i(t)$, the more private stay the interests of peer $Q^i$ vs $DB$. By assimilating $Y^i(t)$ to a random variable and relative frequencies of the queries in $Y^i(t)$ to probabilities, the intuition of query diversity mentioned above can be expressed as maximizing $H(Y^i(t))$. Hence, a reasonable privacy utility function for $Q^i$ is $U(Y^i(t)) = H(Y^i(t))$.

#### *5.2.2 Shannon's entropy at the category level*

It may be argued that maximizing the entropy at the query level does not provide sufficient privacy protection if some of the queries are semantically similar. For example, if the different queries in $Y^i(t)$ are "astronomy", "telescope" and "orion's belt", then, no matter the relative frequencies of those three queries, $Y^i(t)$ indicates an interest in astronomy.

An alternative is to consider a set of broad semantic categories (*e.g.* "Science", "Religion", "Sport", etc.) and compute the entropy at the category level rather than at the query level. That is $U(Y^i(t))$ would be high only if $Y^i(t)$ contains a similar number of queries of each category.

#### *5.2.3 Hierarchical variance*

A natural extension of the categorization idea in the previous section is to consider an ontology of possible queries, like WordNet [33] or ODP [23], as proposed in [15]. By using such an ontology, the queries in $Y^i(t)$ can be hierarchically classified and hence they can be viewed as a sample of a hierarchical nominal categorical attribute. Thus, it is possible to compute the hierarchical variance of $Y^i(t)$. Such hierarchical variance is a measure proposed in [13] and simplified in [12] whose intuitive idea is as follows: a set of nominal values be-

longing to categories which are all children of the same parent category in the hierarchy has smaller variance that a set with children from different parent categories.

We sketch the way to compute the nominal variance in [12], because it is equivalent to [13] and easier to explain. For each query $y \in Y^i(t)$, the so-called marginality $m(y)$ of $y$ is computed as the sum of distances between $y$ and each of the remaining queries in $Y^i(t)$ over the hierarchy tree specified by the taxonomy used; the distance between two queries is computed as the sum of the weights of edges along the shortest path between the two queries, where edges linking nodes at depth $i-1$ in the tree with those at depth $i$ are assigned weight $2^{L-i}$, being $L$ the depth of the taxonomy tree. Then the hierarchical variance of $Y^i(t)$ is defined as the average marginality of the queries in $Y^i(t)$.

If we take the hierarchical variance of $Y^i(t)$ as privacy utility function $U(Y^i(t))$, then $Q^i$ is interested in the strategy providing the largest increase in the hierarchical variance of $Y^i(t)$; this is achieved by submitting new queries which are as distant as possible in the semantic taxonomy from previously submitted queries.

Using ontologies and maximizing the hierarchical variance of queries submitted by a user also helps thwarting the attacks against query bundles described in [18]. It is reported in Section 6.3 below and it can be verified in the AOL query logs [2] that the query interests of real users tend to be rather narrow and focused, and hence with moderate to small hierarchical variance. Therefore, forcing a high hierarchical variance within the ontology implies that the queries originated by a user will be submitted by many different peers, which makes it harder for the database to link with each other the queries originated by the same user. To a lesser extent, the alternative of maximizing category-level entropy suggested in Section 5.2.2 also causes the queries originated by each user to be submitted by several other peers; in this respect, it also provides some protection against the attacks reported in [18].

Of course, vanity queries originated by a user will always be linkable to that user, but this is an intrinsic problem of vanity queries: a privacy-conscious user using query profile obfuscation should in the first place be very cautious about originating vanity queries.

#### *5.2.4 Utility function for location-based queries*

We propose in this section a privacy utility function suited for location-based queries submitted to a commercial provider of location-based services (LBS provider). Such queries are of type "List of services/facilities of type $T$ near location $\ell$", where $\ell$ is typically the user's current location and $T$ can be hotels, pharmacies, restaurants, monuments, etc. In this kind of queries, users want

to keep their location profile private rather than or in addition to keeping their query profile private. In other words, a user does not want the LBS provider to know her exact trajectory.

Most solutions for location-based privacy are standalone and they rely on the user blurring her location and/or supplying an area rather than an exact location (*e.g.* [19,9,8]). However, in [26], a peer-to-peer protocol in which users submit other user's queries to obfuscate their own location profile was presented.

If we want to characterize peer-to-peer location privacy as a rational game, we must assume that each peer wants to maximize her location privacy. A way to do this is for the peer to submit locations to the LBS server which are as uniformly spread as possible.

Assume the space over which locations are taken can be represented as a rectangle. This entails no loss of generality, because even the map of the world can be represented as a rectangle. Assume further that the rectangle of space has been divided into $M$ square tiles (a rectangle can be approximately covered by squares if the latter are small enough).

Consider a peer $Q^i$ having submitted a set of $n_i(t)$ location-based queries. Assume that we discretize locations at the tile level, that is, we regard as equivalent all locations within the same tile. The inaccuracy of the above discretization decreases with the tile size and, to get the smallest possible error, the tile size should be as small as the resolution of the technology used to determine the coordinates of locations. Maximum privacy for $Q^i$ is achieved when the discretized locations corresponding to the submitted queries are uniformly distributed among the tiles, that is, when each of the $M$ tiles contains $n_i(t)/M$ locations (if $n_i(t)/M$ is integer) or a rounded value of $n_i(t)/M$ (if $n_i(t)/M$ is not an integer). This maximum-private set of discretized queries can be represented as the following vector

$$V_P^i = (n_i(t)/M, \cdots, n_i(t)/M)$$

Now, given any set $Y^i(t)$ of $n_i(t)$ submitted location-based queries, such that the $l$-th tile contains $k_l$ locations, we can use the same discretization above and represent $Y^i(t)$ as

$$V^i = (k_1, \cdots, k_M)$$

The distance between vectors $V^i$ and $V_P^i$ is a measure of the privacy offered by $Y^i(t)$: the closer to zero that distance, the more privacy is offered by $Y^i(t)$ to $Q^i$. Hence, a reasonable privacy utility measure is

$$U(Y^i(t)) = \frac{1}{1 + \sum_{l=1}^{M}(k_l - n^i(t)/M)^2}$$

At time $t+1$, $Q^i$ will be interested in submitting a query to the LBS server if doing so causes $U(Y^i(t+1)) > U(Y^i(t))$.

### *5.3 Performance issues*

In order to prevent a message from being forwarded from one peer to the next without being submitted by anyone, the number of hops can be limited by adding a field to the message header specifying the count of allowed remaining hops. Such a limitation has the benefit of bounding the response time and the query exposure (number of peers who see the query). Call this field *hop_count*. A possible adaptative algorithm to dynamically manage this field is as follows.

**Algorithm 1 (Maximum hop count adjustment)**

*(1) The first time a peer $Q^i$ originates a query, say $x_0$, $Q^i$ initializes a hop_count for that query to a random value $hc_0^i$ around $N/2$, where $N$ is the number of peers.*
*(2) If a peer $Q^j$ receives the forwarded $x_0$ with hop_count $> 1$, then $Q^j$ has the usual choice between submitting $x_0$ or further forwarding it. If the choice is to forward, then $Q^j$ decrements the hop_count field by one before forwarding the query.*
*(3) If a peer $Q^j$ receives a forward query with hop_count $= 0$, then $Q^j$ must submit the query.*
*(4) The peer submitting the query adds the current number of remaining hops to the answer before returning the answer to the originator following the reverse path.*
*(5) If the query originator $Q^i$ sees that the remaining hop_count for $x_0$ is 1 or more, the hop_count for the next query $x_1$ she originates will be initialized to $hc_1^i := hc_0^i - 1$; otherwise if the remaining hop_count for $x_0$ is 0, then the hop_count for the next originated query $x_1$ will be initialized to $hc_1^i := \min(hc_0^i + 1, N)$.*
*(6) A similar procedure is followed for the successive queries originated by $Q^i$: if not all hops are exhausted for one query, the next query is allowed one less hop; if all hops are exhausted, the next query is allowed one more hop (with $N$ hops at most). This adapts the delay tolerance to the "mood" of the peers: if they are more helpful, tolerate less hops; if they are less helpful, allow for more hops.*

Two remarks about the above algorithm are in order:

- It is important that the first value $hc_0^i$ is a *random* value around $N/2$: if it were fixed to $N/2$, the first peer receiving the query from $Q^i$ would know that the query had indeed been originated by $Q^i$.

- We assume peers to be honest-but-curious; hence a peer receiving a query with $hop\_count = 0$ can be assumed to submit it. Note that the adaptative algorithm will increase the initial value of *hop_count* for the next query, which should make this "forced submission" situation less likely.

The delay $\tau$ added by the multi-hop game can be computed from the number of hops and the communication and processing time at each hop. Let $n_h$ be the number of hops, $\tau_q$ be the time needed to send the query from one peer to the next, $\tau_p$ be the processing time needed for a peer to decide on query submission or forwarding (*i.e.* the time to compute the privacy utility function) and $\tau_R$ the time needed for a peer to return the query results to the previous peer in the reverse path leading back to the peer having originated the query. Then we have

$$\tau = (\tau_q + \tau_p + \tau_R)n_h \tag{2}$$

Note that we do not count as delay the time needed to actually submit the query to $DB$ and obtain the query results, because this time is also needed in case of direct submission. Let us assume that a typical query consists of up to 40 bytes and a typical query results page of up to 80,000 bytes; further, assume a medium-speed ADSL connection speed of 1 Mbps. In this case, $\tau_q = 0.0003$, $\tau_R = 0.64$ s. The computing time $\tau_p$ can be regarded as negligible compared to communication times. Therefore, for $n_h$ hops, we obtain an additional delay (in seconds)

$$\tau = 0.6403n_h \tag{3}$$

## 6 Simulation results

In this section, we report on experimental results using privacy utilities based on Shannon's entropy at the query level (see Section 5.2.1 above). Several tests were carried out for the multi-hop $N$-peer game using a simulated and a real query generation scenario, with $N > 2$.

For experiments with simulated queries, the time between two successive queries by a peer $Q^i$ was generated by sampling an exponential random variable with parameter $\lambda_i$; hence, the query generation rate of $Q^i$ was $\lambda_i$ queries per time unit and the expected time between successive queries for $Q^i$ was $1/\lambda_i$. For experiments with real AOL queries, the real between-query times were used.

For each experiment, we report the evolution of the average number of hops, which is constrained by Algorithm 1.

### *6.1 Collaboration metrics*

The metrics considered to analyze the (lack of) collaboration among peers were:

- *Directly Submitted Queries (DSQ).* For peer $Q^i$, $DSQ^i(t)$ is the fraction of queries among those originated by $Q^i$ up to time $t$ that were directly submitted to $DB$ by $Q^i$.
- *Forwarded Queries (FQ).* For peer $Q^i$, $FQ^i(t)$ is the fraction of queries among those received from other peers up to time $t$ that $Q^i$ forwards instead of submitting them.

Note that, in general, DSQ and FQ are independent metrics. Indeed, up to time $t$ peer $Q^i$ has asked for help to submit a fraction $1 - DSQ^i(t)$ of her originated queries. The fact that $Q^i$ asks for a certain proportion of help does not determine her own willingness to help others: $Q^i$ can submit any fraction $1 - FQ^i(t)$ of the queries received from other peers. Looking at the overall set of peers, if all of them are very unwilling to help (high FQ for all of them), what will happen is that the number of hops needed for each query will increase, perhaps up to the maximum hop count allowed, but in the end the query will be submitted by some peer. Hence low DSQ (much help requested) is compatible with high FQ (little willingness to provide help) and, of course, high DSQ (not much help requested) is compatible with low FQ (great willingness to provide help).

### *6.2 Results using a simulated environment*

#### *6.2.1 Influence of the query generation rate*

We first simulated an $N$-peer game with $N = 5$, where Peer $i$, for $i = 1$ to 5, had a different query generation rate $\lambda_i$. Otherwise said, the time between successive queries by Peer $i$ was drawn from an exponential distribution with parameter $\lambda_i$ (expected time $1/\lambda_i$). Specifically, we took $\lambda_1 = 0.2$, $\lambda_2 = 0.4$, $\lambda_3 = 0.6$, $\lambda_4 = 0.8$ and $\lambda_5 = 1$. This means that Peer 1 originated less queries in the simulated scenario than Peer 2, Peer 2 less than Peer 3, and so on. Each query value was (uniformly) randomly drawn from a set of 1000 possible queries.

Figure 2 displays the evolution of the DSQ and FQ metrics, as well as the number of hops needed for a query to be submitted, using the scenario described above. The abscissae in this figure and in the following figures in this section are the total number of queries submitted by all peers. The evolution of DSQ shows that peers with a comparatively higher query generation rate

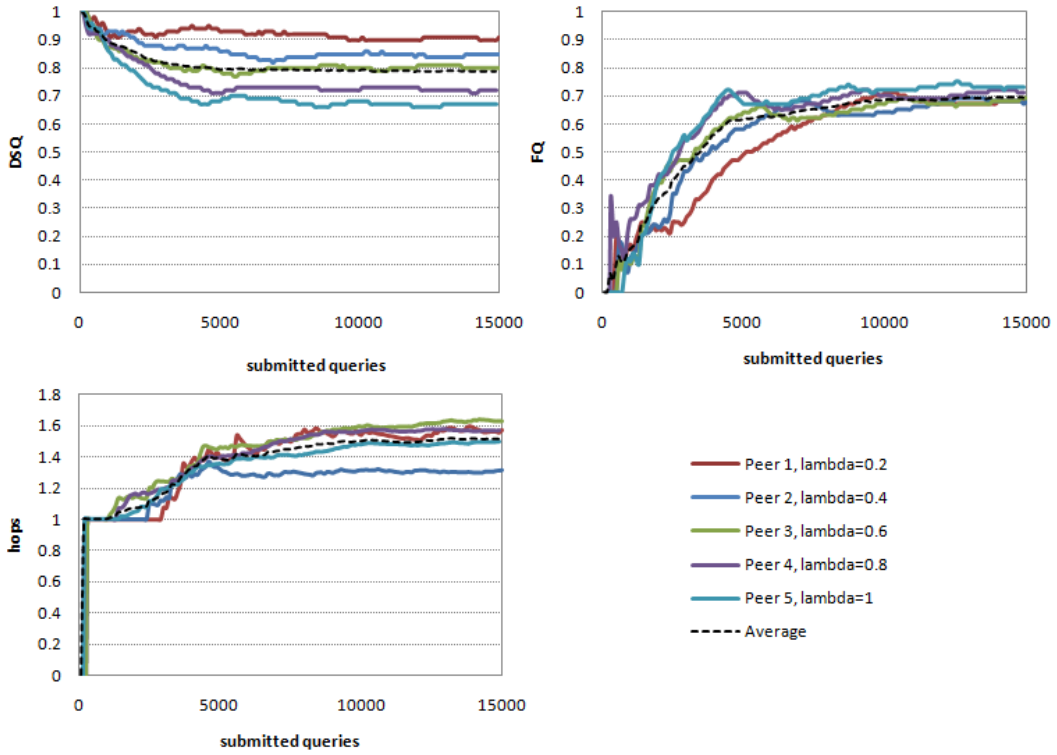


Fig. 2. Evolution of DSQ, FQ and the average number of hops needed to submit a query using a simulated scenario of 5 peers with different query generation rates

tend to submit fewer queries by themselves (lower DSQ) than those peers who originated few queries. At the same time, the FQ evolution shows that peers with a comparatively lower query generation rate tend to forward fewer queries among those received from other peers. The rationale is that peers with a higher query generation rate need more help in submitting queries in order to obfuscate their profile vs $DB$, and peers with a lower query generation rate are eager to help submitting other people's queries in order to hide their own few queries. Hence, this simulation illustrates a Nash equilibrium between players generating more queries and those generating less queries.

It can also be noticed that FQ stabilizes around 70% in the second half of the simulation interval. This happens after the histograms of submitted queries become nearly homogeneous (flat) for all peers: in this situation, peers have little interest in helping other peers, because they are perfectly happy with their flat histograms.

The average number of hops needed by a query before being submitted increases a little as the simulation progresses, but it stabilizes under 2; according to Expression (3), this implies a time delay of about 1 second.

### *6.2.2 Influence of the number of peers*

To examine the influence of the number of peers, we created scenarios with $N = 5, 50$ and 100 peers. Like above, each query value was (uniformly) randomly drawn from a set of 1000 possible queries. Figure 3 displays the evolution of DSQ and FQ, as well as the number of hops that were needed for a query to be submitted. In a scenario with $N = 5$ peers the average DSQ stabilizes around 75%, with $N = 50$ it stabilizes around 65%, and with $N = 100$ it stabilizes around 63%. A higher DSQ for a low number of peers ($N = 5$) can be construed as follows: the less peers, the easier it is that the query originated and forwarded by a peer returns to the originating peer after some hops and the originating peer ends up submitting her own query.

Regarding the average FQ, its behavior seems pretty independent of the number of peers and its average stabilizes around 80% in all cases. Independence of FQ from the number of peers is reasonable, because for each peer FQ measures the peer's unwillingness to help other peers: such an unwillingness depends only on how flat the peer's current query profile vs $DB$ is and it does not depend on how many peers ask for help.

As to the average number of hops, it can be seen that dynamically adjusting *hop_count* as proposed above is efficient: the average number of hops required for submission increases only slightly with the number of peers, and it stabilizes around 3 hops in a scenario of 100 peers. Hence, with 100 peers the time delay introduced is about 2 seconds, according to Expression (3).

### *6.3 Results using real queries*

To study the system in a real environment, we used a mirror of the AOL database [2], which contains twenty million queries from 650000 users over a 3-month period. We randomly picked five users among those having originated at least 1000 queries, with their respective first queries issued on March 1, 2006. These users issued their subsequent queries with very different between-query times, up to May 31, 2006. We took these five users, their query values and query times as if they corresponded to five peers in our system.

Figure 4 shows the DSQ and FQ evolution, as well as the average number of hops a query needs to be submitted, for those five AOL peers. The average number of hops for all peers is always under 2, which implies that the time delay, according to Expression (3), is about 1 second. The evolution of FQ is different for each peer, as it depends on the sequence of queries a peer originates. It can be seen that, on average, DSQ stabilizes around 70% and FQ around 45%. Generally, in this simulation peers are readier to help other peers in submitting their queries to $DB$. This is because in a real environment

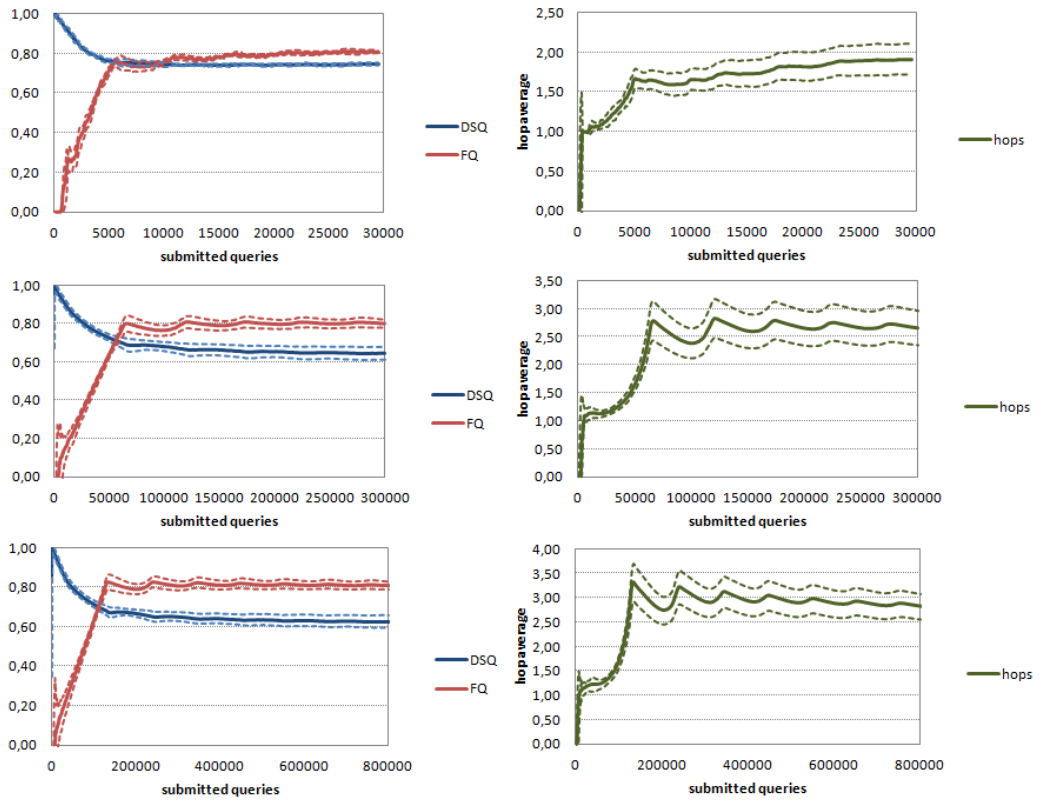


Fig. 3. Evolution of DSQ average, FQ average and the average number of hops needed to submit a query using a simulated scenario with 5 peers (top), 50 peers (middle) and 100 peers (bottom) with the same query generation rate ($\lambda = 1$). Dotted lines represent plus/minus one standard deviation w.r.t. to the average represented by the continuous line of the same color.

each user tends to be interested in a subset of the total queries, and submitting queries on behalf of other peers is a way to hide her own profile. The less diverse the queries originated by a user, the lower is her DSQ (she is less self-sufficient) and her FQ (she is more collaborative); *e.g.* in Figure 4, Peer 2 has probably less diverse interests than the rest, because she forwards a lower proportion of the queries she receives than the other peers (lower FQ), and she directly submits a slightly lower proportion of her originated queries (lower DSQ).

## 7 Conclusions and future research

Multi-hop peer-to-peer profile obfuscation is an attractive option to keep the query profiles of users private. Its strong point is that the help of peers can be leveraged by a user without the user worrying about staying private in front of specific peers: indeed, when a peer receives a query, she does not know who originated it. Potentially long response times and exposure of queries to many

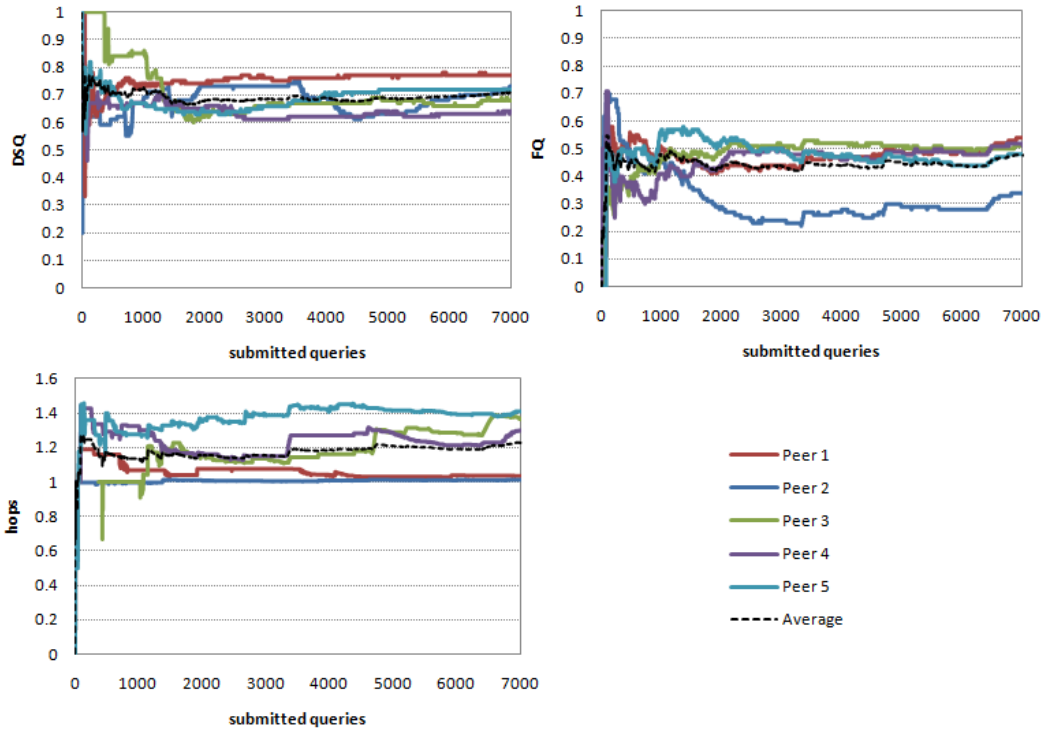


Fig. 4. Evolution of DSQ, FQ and the average number of hops to submit a query, for a set of five peers corresponding to five real users in the AOL database

peers are weaknesses which can be kept at bay if the number of hops is limited using a mechanism like the one we have proposed.

A problem shared by all P2P systems is that they depend on the helpfulness of peers. The game-theoretic algorithms presented in this paper allow peers to decide in an automated and rational way whether and how they collaborate in submitting each other's queries. Several different privacy utility functions have been suggested to capture the peers' privacy interests.

Future research may include at least the following issues:

- *New privacy utility functions.* Depending on the type of queries or the user requirements, other privacy utility functions different from the ones proposed in this paper may be conceivable.
- *Dealing with cheaters.* In this paper, we have assumed peers to be honest-but-curious: they follow the protocol even if they may attempt to find out who is the query originator. A cheater is a peer who does not follow the protocol when she receives a query: cheating may consist in neither submitting nor forwarding the query, or not returning the query answer along the reverse path. This can be detected because the number of hops and hence the maximum turnaround time are limited when all peers behave honestly. If, after a timeout, a query originator $Q^i$ receives no answer to a query $Q^i$ forwarded to $Q^j$, the easiest solution is to retry and forward the query to

some peer $Q^{j'}$ with $j' \neq j$. Retrial will work if the random path followed by the query the second time ends up in a query submitter without having crossed any cheater. Future research could involve finding ways for the query originator to trace and identify cheaters, in order to discourage cheating: cryptographic solutions and/or reward/punishment mechanisms could be envisioned.

## Disclaimer and acknowledgments

The authors are with the UNESCO Chair in Data Privacy, but the views expressed in this paper are their own and do not commit UNESCO. This work was partly funded by the European Commission under FP7 project "DwB", by the Spanish Government through projects TSI2007-65406-C03-01 "E-AEGIS" and CONSOLIDER INGENIO 2010 CSD2007-0004 "ARES", and by the Government of Catalonia under grant 2009 SGR 1135. The first author is partly supported as an ICREA-Acadèmia researcher by the Government of Catalonia.

## References

[1] C. Aguilar-Melchor and Y. Deswarte. Trustable relays for anonymous communication. *Transactions on Data Privacy*, 2(2):101-130, 2009.

[2] "AOL Search Data Scandal", August 2006. `http://en.wikipedia.org/wiki/AOL_search_data_scandal`

[3] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: Pir with preprocessing. *Journal of Cryptology*, 17:125–151, 2004.

[4] J. Castellà-Roca, A. Viejo and J. Herrera-Joancomartí. Preserving user's privacy in web search engines. *Computer Communications*, 32(13-14):1541-1551, 2009.

[5] B. Chor, N. Gilboa and M. Naor. Private information retrieval by keywords. Technical Report TR CS0917, Department of Computer Science, Technion, 1997.

[6] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 41–50, 1995.

[7] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *Journal of the ACM*, 45:965–981, 1998.

[8] R. Di Pietro and A. Viejo. Location privacy and resilience in wireless sensor networks querying. *Computer Communications*, 34(3):515-523, 2011.

[9] J. Domingo-Ferrer. Microaggregation for database and location privacy. In *NGITS 2006*, volume 4032 of *Lecture Notes in Computer Science*, pages 106–116. Berlin-Heidelberg: Springer, 2006.

[10] J. Domingo-Ferrer, M. Bras-Amorós, Q. Wu and J. Manjón. User-private information retrieval based on a peer-to-peer community. *Data and Knowledge Engineering*, 68(11):1237–1252, 2009.

[11] J. Domingo-Ferrer and U. González-Nicolás. Rational behavior in peer-to-peer profile obfuscation for anonymous keyword search. *Information Sciences*, 185(1):191–204, 2012.

[12] J. Domingo-Ferrer and G. Rufian-Torrell. Generalized microaggregation and generation of hybrid nominal and non-nominal data. Manuscript, 2011.

[13] J. Domingo-Ferrer and A. Solanas. A measure of nominal variance for hierarchical nominal attributes. *Information Sciences*, 178(24):4644–4655. 2008. Erratum in *Information Sciences*, 179(20):3732, 2009.

[14] J. Domingo-Ferrer, A. Solanas and J. Castellà-Roca. $h(k)$-Private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, 33(4):720–744, 2009.

[15] A. Erola, J. Castellà-Roca, G. Navarro-Arribas and V. Torra. Semantic microaggregation for the anonymization of query logs. In *Privacy in Statistical Databases-PSD 2010*, eds. J. Domingo-Ferrer and E. Magkos, volume 6344 of *Lecture Notes in Computer Science*, pages 127-137. Berlin-Heidelberg: Springer, 2010.

[16] GoogleSharing, `http://www.googlesharing.net/`

[17] D. C. Howe and H. Nissenbaum. TrackMeNot: Resisting surveillance in web search. In *Lessons from the Identity Trail: Privacy, Anonymity and Identity in a Networked Society*. I. Kerr, C. Lucock and V. Steeves(eds.). Oxford UK: Oxford University Press, 2009. Software downloadable from: `http://www.mrl.nyu.edu/~dhowe/trackmenot/`

[18] R. Jones, R. Kumar, B. Pang and A. Tomkins. Vanity fair: privacy in querylog bundles. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management-CIKM 2008*. New York, NY: ACM, 2008.

[19] M. F. Mokbel. Towards privacy-aware location-based database servers. In *Proceedings of the Second International Workshop on Privacy Data Management-PDM 2006*, Atlanta GA, 2006. IEEE Computer Society.

[20] M. Murugesan and C. Clifton. Providing privacy through plausibly deniable search. In *2009 SIAM International Conference on Data Mining-SDM 09*, Sparks NV, USA, 2009.

[21] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54:289-295, 1951.

[22] N. Nisan, T. Roughgarden, É. Tardos and V. V. Vazirani (eds.). *Algorithmic Game Theory*. Cambridge University Press, 2007.

[23] Open Directory Project. `http://www.dmoz.org`

[24] W. Ogata and K. Kurosawa. Oblivious keyword search. *Journal of Complexity*, 20(2-3): 356-371, 2004.

[25] R. Ostrovsky and W. E. Skeith-III. A survey of single-database PIR: techniques and applications. In *Public Key Cryptography-PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411, Berlin Heidelberg, 2007.

[26] D. Rebollo-Monedero, J. Forné, A. Solanas and A. Martínez-Ballesté. Private location-based information retrieval through user collaboration. *Computer Communications*, 33(6):762-774, 2010.

[27] M. K. Reiter and A. D. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information Systems Security*, 1(1):66-92, 1998.

[28] C. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27(July and October):379-423 and 623-656, 1948.

[29] K. Stokes and M. Bras-Amorós. On query self-submission in peer-to-peer user-private information retrieval. In *Proc. of the 4th International Workshop on Privacy and Anonymity in the Information Society-PAIS 2011*, pp. 7:1-7:5, ACM Digital Library, 2011.

[30] The Tor Project, Inc. "Tor: Overview". `http://torproject.org/overview.html.en`

[31] Torbutton 1.2.5. `https://www.torproject.org/torbutton`

[32] A. Viejo and J. Castellà-Roca. Using social networks to distort users' profiles generated by web search engines. *Computer Networks*, 54(9):1343-1357, 2010.

[33] "WordNet: A lexical database for English", Princeton University. `http://wordnet.princeton.edu`