Rational Behavior in Peer-to-Peer Profile Obfuscation for Anonymous Keyword Search

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

User profiling in web search has the advantage of enabling personalized web search: the quality of the results offered by the search engine to the user is increased by taking the user's interests into account when presenting those results. The negative side is that the interests and the query history of users may contain information considered as private; hence, technology should be provided for users to avoid profiling if they wish so. There are several anti-profiling approaches in web search, from basic level countermeasures to private information retrieval and including profile obfuscation. Except private information retrieval (PIR), which hides the retrieved item from the database, the rest of approaches focus on anonymizing the user's identity and fall into the category of anonymous keyword search (also named sometimes user-private information retrieval). Most current PIR protocols are ill-suited to provide PIR from a search engine or large database, due to their complexity and their assumption that the database actively cooperates in the PIR protocol. Peer-to-peer profile obfuscation protocols appear as a competitive option provided that peers are rationally interested in helping each other. We present a game-theoretic analysis of P2P profile obfuscation protocols which shows under which conditions helping each other is in the peers' rational interest.

Key words: Anti-profiling; Private information retrieval; Anonymous keyword search; User-private information retrieval; Privacy-preserving data mining; Game theory.

1 Introduction

According to [16], by the end of 2010 more than 28.7% of the world's population had access to Internet and, according to [21], in May 2011 there were 324 million active web sites. Accessing this huge volume of information without the help of search engines (*e.g.* Google, Yahoo, etc.) is not conceivable. Web search engines (WSEs) facilitate the search for information about one or several terms and they return several pages of search results consisting of links to web pages with information containing the searched terms.

It is mentioned in [17] that 68% of users click on a search result in the first page and 92% in the first three pages. For this reason, it is very important that WSEs place the links that are most interesting to users in the first result pages. This leads to the problem of finding out what are the interests of users. For example, for a user interested in astronomy, a search for "mercury" should ideally return links on the planet at the top, while, for a user interested in chemistry, links at the top should correspond to the chemical element. Every user has her own interests, and these may change over time.

The common situation in WSEs is that users do not collaborate with the WSE by sending information on their interests beyond the searched keywords. Hence, the WSE may try to infer those interests using the browsing history, click-through data, web community information, or a client-side application storing the user's interests. The most common and successful approach for web search engines is to create user profiles based on the queries previously submitted by the users. This profiling approach is known as personalized web search and its positive side is an increase of the quality of the results offered to the user. The cost of profile creation by the WSE is more than compensated by the possibility to embed in the search results advertising tailored to the user's interests.

The downturn of user profiling is that the interests and the query history of users may contain information considered as private. For example, if a user has looked up a certain disease, it can be inferred that either the user or someone close to her suffer from that disease. This is not an academic speculation: in 2006, 20 million queries submitted by 658000 users of AOL were publicly disclosed; AOL claimed queries to be properly protected against user re-identification. Two *New York Times* journalists identified a user after studying the released queries [2]. What has been said so far about profiling by search engines also applies to any queryable database whose users do not want their interests to be linked to their identities [10].

1.1 Contribution and plan of this paper

As argued in the section on related work below (Section 2) 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. 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 P2P profile obfuscation system.

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 reasonable because the receiver does not see the forwarder's complete query profile: with N peers in the community, the receiver just sees on average $1/N$-th of the forwarder's profile. This thwarts re-identification by profile. Even if one particular query can be re-identified (*e.g.* a vanity query), it will be linkable (on average) only to $1/N$-th of the corresponding user's profile. If this possibility or re-identification by IP address among peers are a concern, a system like Tor [30] might be used to provide transport-level anonymity in peer-to-peer communication; also, Tor would protect against timing attacks, because it would introduce some delay uncorrelated to the topology of the P2P community.

The contributions in this article are as follows. We give a game-theoretic analysis of *single-hop* P2P profile obfuscation systems, like [12,5]. We specify a metric for the privacy of peers vs the database and vs other peers. We compute the privacy utilities of the different strategies that can be followed by the peers in view of maximizing their privacy. Maximizing that privacy utility for 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 [20,22] 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 database.

This paper focuses only on rational behavior in single-hop systems, but it does not deal with multi-hop systems, where the query may pass through several intermediate peers before being submitted. We next give some justification. In a single-hop system, the two peers exchanging a query learn each other's decisions because they see the outcome of those decisions: if A sends a query to B for submission, A will either receive a response from B (implying that B submitted the query) or A will receive nothing (implying that B rejected

the query); also, B knows that the received query was generated by A. Such complete knowledge allows peers to know at all times how private they stay vs each other and vs the database, and thereby make rational decisions. In a multi-hop system, 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. For example, a peer B who receives a query from a peer A does not know whether the query was originated or merely forwarded by A. Similarly, if A generates and forwards a query to B, A cannot predict whether B's rational decision will be to submit the query or forward it. Thus, with such incomplete knowledge, it is hard for peer A (resp. peer B) to make rational decisions. Worse yet, if we mentioned in Section 2.3.3 that in multi-hop systems like Tor, Crowds or [32] the average response time was quite long, in a *rational* multi-hop system there is not even any guarantee of a bounded response time: a peer having received a query may always rationally decide to further forward it to another peer rather than submitting it. Even if a timeout is enforced, the peer who happens to be dealing with the query when the timeout elapses may refuse to submit the query, so that the query generator still does not get any response within the timeout.

Section 2 describes related work. Section 3 gives some background on game theory. Section 4 proposes a characterization of a P2P profile obfuscation game. Simulation results are reported in Section 5. Section 6 summarizes conclusions and future research lines.

2 Related work

We briefly review the literature on preventing user profiling in web and database search. Partly inspired in [29], we define four levels of privacy in web search: *basic level*, *pseudo-identity*, *profile obfuscation*, and *private information retrieval*. Except private information retrieval (PIR), which hides the retrieved item from the database, the rest of approaches focus on anonymizing the user's identity and fall into the category of anonymous keyword search (named sometimes user-private information retrieval, *e.g.* [12]).

2.1 Basic level

Using dynamic IP addresses and a web browser rejecting cookies is the simplest approach to avoid profiling. Unfortunately, this approach is often hard to follow:

- The renewal policy of the dynamic IP address is not controlled by the

user, but by the network provider, who might always assign the same IP to the same Media Access Control (MAC) address. Also, certain users require static IP addresses.
- Regarding cookie rejection, it implies an unacceptable loss of functionality in many web applications, which require cookies to be enabled on the user's browser. A pragmatic approach is discretionary cookie rejection, whereby only cookies from certain domains are accepted or rejected.

2.2 Pseudo-identity

In this second level, the user's identity is replaced by a pseudonym or a privacy-preserving certificate [18]. The WSE can still create profiles associated to pseudonyms. As queries in a profile accumulate, their sensitive information makes it easier to determine the real identity to which the profile corresponds. Hence, the privacy level is low: *e.g.* AOL replaced IP addresses with pseudo-identities in their query logs and this did not prevent re-identification [2].

2.3 Profile obfuscation

In this privacy level, the WSE is unable to exactly profile single users. This is good for privacy protection, but it reduces the accuracy of web search. At least three profile obfuscation strategies can be conceived: standalone, proxy-based and peer-to-peer.

2.3.1 Standalone obfuscation

A user obfuscates her profile without external help. In [13] a system named GooPIR is proposed where a user masks her target query by locally ORing it with $k-1$ fake queries and then submits the resulting masked query to a search engine or large database which does not need to cooperate (in fact, it does not even need to know that the user is trying to protect her privacy). This system works fine but it assumes that the frequencies of keywords and phrases that can appear in a query are known and available, *e.g.* from a Thesaurus: for maximum privacy, the frequencies of the target and the fake queries should be similar.

An approach with the same spirit of GooPIR was independently and subsequently proposed in [19]. This approach is based on the so-called plausibly deniable privacy. A user issuing a set of queries $S = \{Q_1, \cdots, Q_k\}$, where Q_i is the user's real target query, has k-plausibly deniable privacy of Q_i if: i) the user can show that any query $Q_j \in S$ would have generated the set S with

equal probability; ii) all $Q_j \in S$ are on different topics, and iii) all $Q_j \in S$ are equally plausible as actual user queries. When a user wants to submit a target query, the system uses a latent semantic indexing based approach to generate $k-1$ cover queries satisfying the previous three conditions. Note that the idea of equal plausibility of cover queries and target query is similar to the idea in GooPIR of taking the $k-1$ fake queries and the target query as having similar frequencies in a Thesaurus.

The disadvantage of the previous two systems is that generating the $k-1$ cover queries is not straightforward: either a Thesaurus is needed (GooPIR) or queries must be semantically processed [19]. TrackMeNot [15] is another practical system based on a different principle: rather than submitting a single masked query for each actual query like GooPIR or [19], a browser extension installed in the user's computer hides the user's actual queries in a cloud of automatic "ghost" queries submitted to popular search engines at different time intervals. While practical at a small scale, if the use of TrackMeNot became generalized, the overhead introduced by ghost queries would significantly degrade the performance of search engines and communication networks. Also, the submission timing of automatic ghost queries may be distinguishable from the submission timing of actual queries, thereby providing an intruder with clues to identify the latter type of queries (this weakness of TrackMeNot was also noted by [19]). Finally, since queries often consist of phrases rather than single keywords (*e.g.* see queries in the AOL data set, [2,4]), it may be easy to tell the real phrases submitted by the user from the artificial phrases in the ghost queries.

2.3.2 Proxy-based obfuscation

GoogleSharing [14] is a system consisting of a custom proxy and a Firefox Addon. The proxy works by generating a pool of GoogleSharing "identities", each of which contains a cookie issued by Google and an arbitrary User-Agent for one of several popular browsers. The Firefox Addon watches for requests to Google services from the user's browser, and, when enabled, transparently redirects all of them to a GoogleSharing proxy. There the user's request is stripped of all identifying information and replaced with the information from a GoogleSharing identity.

Additionally, since Google now accepts receiving SSL-encrypted queries, GoogleSharing does not need to see the actual queries submitted by the user, which can travel encrypted from the user's browser up to Google. Furthermore, GoogleSharing claim in their privacy policy that they do not monitor, record or log any user traffic.

The negative point is that a collusion of GoogleSharing and Google is suffi-

cient to reconstruct the query profile of any user. Hence, the users must trust GoogleSharing not to collude with Google and GoogleSharing becomes a *de facto* trusted third party. As we will see below, collusions in P2P profile obfuscation are much more difficult, because the user's profile is diffused among a number of peers.

2.3.3 Peer-to-peer obfuscation

A user obfuscates her profile with the help of a peer-to-peer (P2P) community. A user submits queries on behalf of her anonymous peers and conversely. It can be argued that users possibly prefer being profiled by a search engine to entrusting their peers (friends, colleagues, etc.) with their private queries. While this may be true in democratic countries where search engines are legally bound to privacy preservation, P2P obfuscation may be preferable in non-democratic countries where search engines are government-controlled; it is probably safer for opposers to the regime to help each other to obfuscate their query profiles, because if search engines leak suspect user profiles to the government, individuals re-identified from those profiles might be punished.

Indeed, with P2P obfuscation, in general no one (neither the search engine nor any single peer) sees the complete query profile of a user, which is distributed among the peers. Even if re-identification of query originators via IP addresses is attempted, P2P obfuscation provides additional protection: without P2P obfuscation, only the IP addresses of those submitting to the search engine need to be tracked (because the query submitters are the query originators); however, with P2P obfuscation, the query submitters are not the query originators, so that, to re-identify the originators, one needs to track all IP traffic between peers. Of course, a non-democratic regime could make the very coarse decision of considering the entire set of query submitters as opposers and attempt action against all of them; even in that case, though, query originators could resort to foreign query submitters outside the regime's jurisdiction.

A relatively easy P2P option is to use an onion-routing system like Tor [30] to anonymously route queries in several hops from the originating user to the WSE; the query results are returned following the reverse path. Here the peer nodes merely act as routers. Such a system anonymizes the transport of data, but it gives no end-to-end protection (at the application level). As long as the WSE can link the successive queries submitted by the same user (*e.g* by using cookies), it can profile and re-identify the user. Torbutton [31] is a browser plug-in that complements Tor by providing the aforementioned application-level protection and thereby enabling P2P profile obfuscation. Anyway, this solution shares with other multi-hop solutions mentioned below a rather long response time: with a limit of two hops, Tor with Torbutton is 20 times slower

than a direct connection and it takes about 10 seconds to get a query response [27]; with the default three hops of Tor, it can easily take 1 minute or more.

The system proposed in [12] uses memory sectors, which are shared by a group of users; information recorded in the sector is encrypted using a symmetric encryption key shared by the group of users sharing the sector. When a user in the group needs the help of another user in the group to submit a query, the first user records the encrypted query in the sector; then some other user retrieves the query, decrypts it, submits it and records the encrypted query results in the sector, where they can be recovered by the first user. This approach certainly has some advantages: unlike [13], it requires no knowledge of the frequencies of all possible keywords and phrases that can be queried; unlike [15], it avoids the overhead of ghost query submission. In addition to preserving the privacy of a user's query profile in front of the database and external intruders, the system in [12] offers privacy vs peer users, because peers are anonymous to each other.

In [5], another P2P approach to profile obfuscation is proposed. Here anonymity is achieved with the help of a central node which puts in touch a group of n users who want to submit a query. The users execute an anonymous query retrieval protocol whereby each user may get a query from one of the remaining $n-1$ users, without knowing what user the query comes from. Each user submits the received query to the database, and she broadcasts to all other users the answer obtained from the database. In this way, the user who originally issued the query can pick up the corresponding answer.

In [32], a P2P profile obfuscation system is presented which rests on a social network. The system is multi-hop in that a user A issuing a query forwards it to one of his relations, say B, in the social network; B may either submit the query directly to the database or forward it to one of her relations, say C; and so on until a node submits the query. The answer to A's query is returned to A by following the reverse path. The goal is to have the queries by each user A as evenly submitted by other users as possible, thereby obtaining maximum privacy in front of the database. An attractive feature of this system is that an existing social network is used as a peer community, which makes deployment easier.

Crowds [26] can provide multi-hop P2P profile obfuscation in a way similar to [32]. The advantage of Crowds is that it is based on a flexible, general system. A shortcoming shared with [32] is that intermediate nodes must be trusted not to leak information, especially the first intermediate node. Another shortcoming specific to Crowds is that, since a fixed forward probability is used, the user originating the query has the highest probability of submitting it. Therefore, if the same query is submitted several times, a good guess for

the WSE is to link it to the user having submitted it more times.

2.4 Private information retrieval

The objective of PIR 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 is the most ambitious privacy level. In the PIR literature (see the seminal papers [6,8] and the more recent [25,1]) the database is modeled as a vector, and one assumes that the user wishes to retrieve the i-th component of the vector while keeping the index i hidden from the database. A more flexible form is keyword PIR [7,23], in which the user can submit a query consisting in a keyword.

PIR protocols proposed so far have two fundamental shortcomings which hinder their practical deployment:

- If the database contains n items, theoretical PIR protocols have $O(n)$ complexity [6,8]: 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].
- 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; actually, PIR is likely to be unattractive to most companies running queryable databases, as it limits their profiling ability.
- 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 does not really fit, as it still assumes a mapping between individual keywords and physical addresses (in fact, each keyword is used as an alias of a physical address). A search engine allowing only searches of individual keywords stored in this way would be much more limited than real engines like Google or Yahoo.

Since the above PIR shortcomings seem quite serious, profile obfuscation seems to be the best practical anti-profiling alternative.

3 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 give, 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.

4 The P2P profile obfuscation game

Consider a system with N players $Q^1, \cdots, Q^N$, and a 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;
Sij: Q^i forwards her query to Q^j, for some $j \neq i$, and requests Q^j to submit the query on Q^i's behalf.

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

Tji: Q^j submits Q^i's query to DB and returns the answer to Q^i;
Tjj: Q^j ignores Q^i's query and does nothing.

We assume that peer Q^j is *honest-but-curious* in the sense that:

- If Q^j chooses Tji, then Q^j returns to Q^i the correct answer obtained from DB;
- Similarly, if Q^j chooses Tjj, then Q^j returns no answer to Q^i (rather than returning a false answer);
- Q^j may read and store Q^i's queries and answers.

4.1 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.

For each query x^i_r in $X^i(t)$, define $F^i(x^i_r, t)$ as the set of players to whom Q^i has forwarded x^i_r for submission up to time t. The players in $F^i(x^i_r, t)$ can be associated relative frequencies as follows: for $j = 1$ to N with $j \neq i$, let $f^{ij}(x^i_r, t)$ be the relative frequency with which Q^i has forwarded x^i_r to player Q^j, up to time t.

The privacy utility function for Q^i should reflect the following intuitions: (i) the more homogeneous the relative frequencies of queries in $Y^i(t)$, the more private stay the interests of Q^i vs DB; (ii) the more homogeneous the relative frequencies of peers in $F^i(x^i_r, t)$ for every $x^i_r \in X^i(t)$, the more private stay the interests of Q^i vs the other peers.

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 $p_1 = \cdots = p_n = 1/n$.

By assimilating $Y^i(t)$ and $F^i(x_r^i, t)$ to random variables and relative frequencies to probabilities, intuition (i) above can be expressed as maximizing $H(Y^i(t))$ and intuition (ii) as maximizing $H(F^i(x_r^i, t))$ for all $x_r^i \in X^i(t)$. Hence, those Shannon entropies are reasonable utility functions for Q^i. When Q^i generates a query x_r^i at time $t+1$:

- Q^i chooses Sii (direct submission) if $H(Y^i(t+1)) \geq H(Y^i(t))$, where $Y^i(t+1) = Y^i(t) \cup \{x_r^i\}$;
- Otherwise Q^i chooses Sik (forwarding the query to Q^k), where

$$k = \arg \max_{j \in \{1,\cdot,N\} \setminus \{i\}} H(F^i(x_r^i, t+1)). \tag{2}$$

In plain words, if direct submission decreases privacy vs DB, the query is forwarded to the player Q^k vs whom the privacy loss is minimum. Note that if Q^i forwards her query to a player Q^j, Q^i always incurs some privacy loss vs Q^j, because Q^j knows the query has been generated by Q^i. Therefore, the best policy is to distribute the successive submissions of a certain query x_r^i as evenly as possible among the various peers. This is what the choice of k in Expression (2) attempts.

When Q^k receives x_r^i, it proceeds as follows:

- Q^k chooses Tki (submitting x_r^i) if $H(Y^k(t+1)) > H(Y^k(t))$, where $Y^k(t+1) = Y^k(t) \cup \{x_r^i\}$;
- Otherwise Q^k chooses Tkk (ignoring x_r^i).

In plain words, Q^k submits x_r^i only if doing so increases her privacy vs the DB.

Consider now the general case where not all peers are viewed as equally privacy-critical by Q^i. Let α_{ij} be a weight in $(0, 1]$ denoting how critical is for Q^i privacy vs Q^j: the higher α_{ij}, the more reluctant is Q^i to forward queries to Q^j. A way to accomodate privacy criticalities is to re-define the relative frequencies as

$$f^{ij}(x_r^i, t) := \frac{\alpha_{ij} c^{ij}(x_r^i, t)}{\sum_{l \in \{1,\cdot,N\} \setminus \{i\}} \alpha_{il} c^{il}(x_r^i, t)} \text{ , for } j \in \{1, \cdots, N\} \setminus \{i\} \tag{3}$$

where $c^{ij}(x_r^i, t)$ is the number of times x_r^i has been forwarded to Q^j up to time t. Clearly, the goal of maximizing $H(F^i(x_r^i, t))$ by making the frequencies $f^{ij}(x_r^i, t)$ in Expression (3) homogeneous implies that x_r^i will be forwarded less often to those players Q^j with higher α_{ij}.

When Q^i chooses strategy Sij to submit a query x_r^i at time $t+1$, since Q^i does not know the utilities of the rest of peers, Q^i must use the trial-and-error procedure described in Algorithm 1 to find the best peer Q^k. In the algorithm,

Q^i identifies the candidate best player Q^k as the player such that asking that player for help maximizes the utility of Q^i (Expression (2)). Then Q^i asks Q^k to submit her query; if Q^k accepts, the algorithm ends; if Q^k refuses, then Q^i excludes Q^k from the set of available players, identifies a new candidate best player $Q^{k'}$ among the remaining available players, and tries $Q^{k'}$; the algorithm goes on until a number of peers equal to the maximum number of attempts *max_attempts* has been tried. If no peer wants to help, then Q^i must submit her query herself.

We next discuss the rationale of restricting the number of tried peers to *max_attempts*. Observe that each candidate best peer Q^k unsuccessfully tried in Algorithm 1 implies some privacy loss, because the originated query is revealed to Q^k. This is accounted for by the update of $f^{ik}(x_r^i, t+1)$, which influences the computation of the next best peer candidate $Q^{k'}$. Hence, systematically trying all possible $N-1$ peers in strategy Sik may entail a loss of privacy for Q^i vs those peers worse than the loss of privacy vs DB that would have been caused by Sii. We control the number of tried peers through the *max_attempts* parameter. The first time Q^i runs Algorithm 1, she will set *max_attempts* := 1 (minimum value). In case of failure in getting help, Q^i resorts to direct submission and increases *max_attempts* by 1; the idea is to adapt to a situation of little peer helpfulness and try to compensate in the next execution the privacy loss of Q^i vs DB caused by (unwanted) direct submission in the current execution. In case of success in getting help in less than *max_attempts* attempts, the algorithm decreases *max_attempts* by 1: since Q^i has easily obtained help in this execution, she can take the risk of allowing less attempts next time. If help has been obtained after exactly *max_attempts*, this value is not changed.

The time complexity of this algorithm is therefore $O(N)$, but on average substantially less than N players will need to be tried (see experimental results in Section 5). The delay τ incurred by the algorithm can be expressed as

$$\tau = (\tau_{id} + \tau_q + \tau_p + \tau_r) * (n_a - 1) + (\tau_{id} + \tau_q + \tau_p + \tau_R) \tag{4}$$

where n_a is the actual number of attempts made, τ_{id} is the processing time needed to identify a candidate best peer (utility computations, up to N entropies), τ_q is the time needed to send the query to the candidate best peer, τ_p is the processing time needed by the candidate best peer to decide on query acceptance or rejection (utility computation, one entropy), τ_r is the time needed for the candidate best peer to communicate rejection (alternatively, it may be a timeout whose expiration means rejection), and τ_R is the time needed for the candidate best peer accepting the query to return the query results to the requesting peer. 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, $\tau_q = 0.0003$ s, $\tau_R = 0.64$ s. If rejection is communicated then $\tau_r \approx \tau_q = 0.0003$ s; if rejection is learnt via timeout, then we need to take, say, $\tau_r = 1$ s. The computing times t_{id} and t_p can be regarded as negligible compared to communication times. Therefore, for n_a attempts, we obtain a delay

$$\tau = 0.0006 * (n_a - 1) + 0.6403 \text{ s} \tag{5}$$

if rejection is communicated and

$$\tau = 1.0003 * (n_a - 1) + 0.6403 \text{ s} \tag{6}$$

if rejection must be learnt via timeout.

Note. Variants of the above control mechanism on *max_attempts* can be conceived. One option would be to set *max_attempts* to a constant value. Another option would be to use the above mechanism but enforce an upper bound less than $N-1$ on *max_attempts* (*e.g. max_attempts* can vary between 1 and, say, 3).

Algorithm 1 BEST PEER SEARCH BY Q^i

$ND := \{Q^1, \cdots, Q^N\} \setminus \{Q^i\}$ {All peers can be candidates}
Read *max_attempts* {Maximum no. of peers to be tried in this execution, $1 \leq$ *max_attempts* $\leq N-1$; in first execution *max_attempts* $:= 1$}
submitted $:= 0$ {$|ND|$ is the cardinality of ND}
while (*submitted* $= 0$) and ($N - 1 - |ND| <$ *max_attempts*) **do**
  Compute $Q^k := \arg \max_{Q^j \in ND} H(F^i(x_r^i, t+1))$
  Send x_r^i to Q^k;
  if Q^k declines to submit x_r^i **then**
    Exclude Q^k from ND {Q^k declines if $H(Y^k(t+1)) \leq H(Y^k(t))$}
  else
    submitted $= 1$
  end if
end while
if (*submitted* $= 1$) and ($N - 1 - |ND| <$ *max_attempts* $- 1$) **then**
  max_attempts $:=$ *max_attempts* $- 1$ {Success in getting help in less than *max_attempts*; next time one less peer will be tried}
end if
if *submitted* $= 0$ **then**
  Q^i submits x_r^i {Q^i submits her own query if no peer wants to help}
  max_attempts $:= \min(N-1, $ *max_attempts* $+ 1)$ {Failure in getting help; next time one more peer will be tried}
end if
Write *max_attempts* {Write *max_attempts* for recovery in the next algorithm execution}

Proposition 1 *If the best peer that Q^i finds for submitting a query x_r^i at time $t+1$ is $Q^k \neq Q^i$ and $H(Y^k(t+1)) > H(Y^k(t))$, where $Y^k(t+1) = Y^k(t) \cup \{x_r^i\}$, then $(Sik(t+1), Tki(t+1))$ is a Nash equilibrium for the game between Q^i and Q^k at time $t+1$.*

Proof: If a best peer $Q^k \neq Q^i$ is being sought, it is because $U_{Sii}(t+1) < U_{Sik}(t+1)$. If $H(Y^k(t+1)) > H(Y^k(t))$ then $U_{Tki}(t+1) > U_{Tkk}(t+1)$, and the proposition follows. □

The above situation in which the best strategy for Q^i is to request Q^k for help and the best strategy for Q^k is to provide that help is called *coprivacy*. See [9,11] for a presentation of the coprivacy theory and its applications.

5 Simulation results

In this section, we report on experimental results using the entropy-based privacy utilities defined in Section 4.1 above. Several tests were carried out for the 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 λ_i; hence, the query generation rate of Q^i was λ_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.

The experimental results show that the number of attempts to find a best peer is at most 5 and it tends to stabilize at lower levels. Therefore, with the parameters used to obtain Expressions (5) and (6), the delay caused by the best peer search is at most 0.6427 seconds if query rejections are explicitly communicated to the requesting peer and 4.6415 seconds if rejections must be learnt by the requesting peer via timeout. This seems an affordable delay to obtain privacy, especially if no timeouts are used.

5.1 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 generated by Q^i up to time t that were directly

submitted to DB by Q^i.

- *Rejected Queries (RQ).* For peer Q^i, $RQ^i(t)$ is the fraction of queries among those received from other peers up to time t that Q^i refused to submit.

5.2 Results using a simulated environment

5.2.1 Influence of the number of peers

We first simulated an N-peer game with $N = 5, 50$ and 100 in order to observe the system's behavior with several numbers of users. Queries were generated by the peers with exponential times with $\lambda = 1$ between successive queries. Each query value was (uniformly) randomly drawn from a set of 1000 possible queries. We assumed that all peers set privacy criticalities for all other peers to 1. The maximum number of attempts was initially fixed to $N - 1$.

Figure 1 displays the evolution of the DSQ and RQ metrics and the number of attempts a peer needs to find the best peer, for $N = 5$ peers with the same query generation rate $\lambda = 1$ and all privacy criticalities set to 1. It can be seen that DSQ stabilized between 67% and 78% (the average was around 75%), whereas the average RQ increased from 0 to 75% until it stabilized close to 72%. The rationale is that peers are more reluctant to help after they achieve a good privacy utility for themselves. The average number of attempts the system makes to find the best peer had a peak at the beginning and then it stabilized below 1. This means that, after a first stage of stabilization, requested queries were submitted on the first attempt.

Figures 2 and 3 display the system's behavior in a game with $N = 50$ and $N = 100$ peers, respectively. To show the DSQ and RQ evolution in a distinct way, we have randomly chosen five peers among the total peers in the system. Results are similar in both cases: the average DSQ stabilized around 70%, and the average RQ increased until it stabilized around 85%. The DSQ metric was a little lower than in the above simulation with 5 peers because in these simulations there were more peers available to forward a query. However, the RQ metric was higher than with 5 peers, because of the large number of queries a peer received to be submitted in behalf of other peers. Remember that a peer only accepts to submit a query received from another peer if submitting it increases the entropy of her overall submitted queries. Regarding the average number of attempts to find the best peer, the evolution was similar: with $N = 50$, the attempts increased up to a peak of 3 attempts (this means a query was being shown to 6% of the peers), and then decreased until they stabilized around 2 (4% of peers); with $N = 100$, the attempts increased up to a peak of 5 attempts (5% of peers), and then decreased until they stabilized around 2.5 (2.5% of peers).

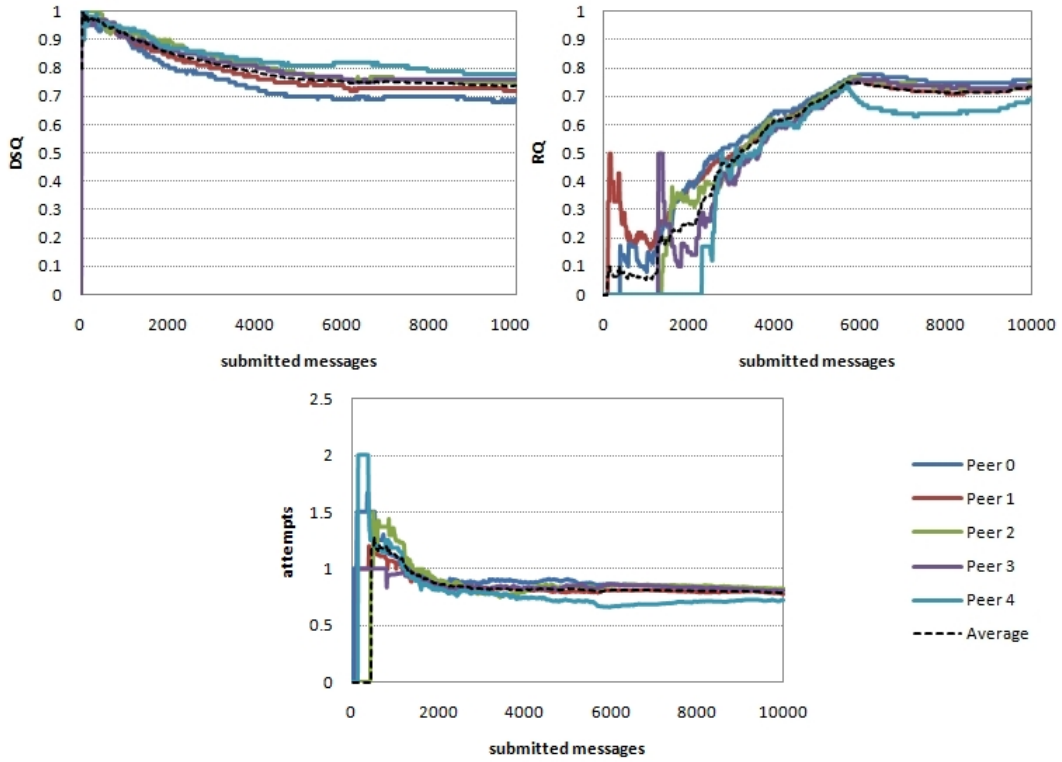


Fig. 1. Evolution of DSQ, RQ and the number of attempts to find the best peer using a simulated scenario with 5 peers having the same query generation rate ($\lambda = 1$) and all privacy criticalities set to 1.

5.2.2 Influence of the privacy criticalities

To examine the influence of the privacy criticalities, we simulated an N-peer game with $N = 5$ using various privacy criticalities. Like above, each query value was (uniformly) randomly drawn from a set of 1000 possible queries. Figure 4 displays the evolution of DSQ and RQ, as well as the average number of attempts a peer needed to find the best peer, when the five peers had different privacy criticalities: $\alpha_0 = 0.2$, $\alpha_1 = 0.4$, $\alpha_2 = 0.6$, $\alpha_3 = 0.8$ and $\alpha_4 = 1$. For simplicity, we assume that the criticality of a peer is the same vs all other peers, that is, Peer 0 is assigned criticality $\alpha = 0.2$ by all other peers, Peer 1 is assigned $\alpha = 0.4$ by all other peers, and so on. The evolution of DSQ shows that those peers who are perceived as less privacy-critical find more help: specifically, Peer 0 with $\alpha = 0.2$ has a substantially lower DSQ than the rest. Regarding RQ and the number of attempts, their evolution does not clearly show the influence of privacy criticalities, because the RQ of a peer is computed as the fraction of queries rejected to *all* other peers; however, it can be seen that the peer perceived as most privacy-critical by everyone else, that is, Peer 4, tends to have a slightly lower RQ (Peer 4 rejects less because all other peers send less requests to Peer 4).

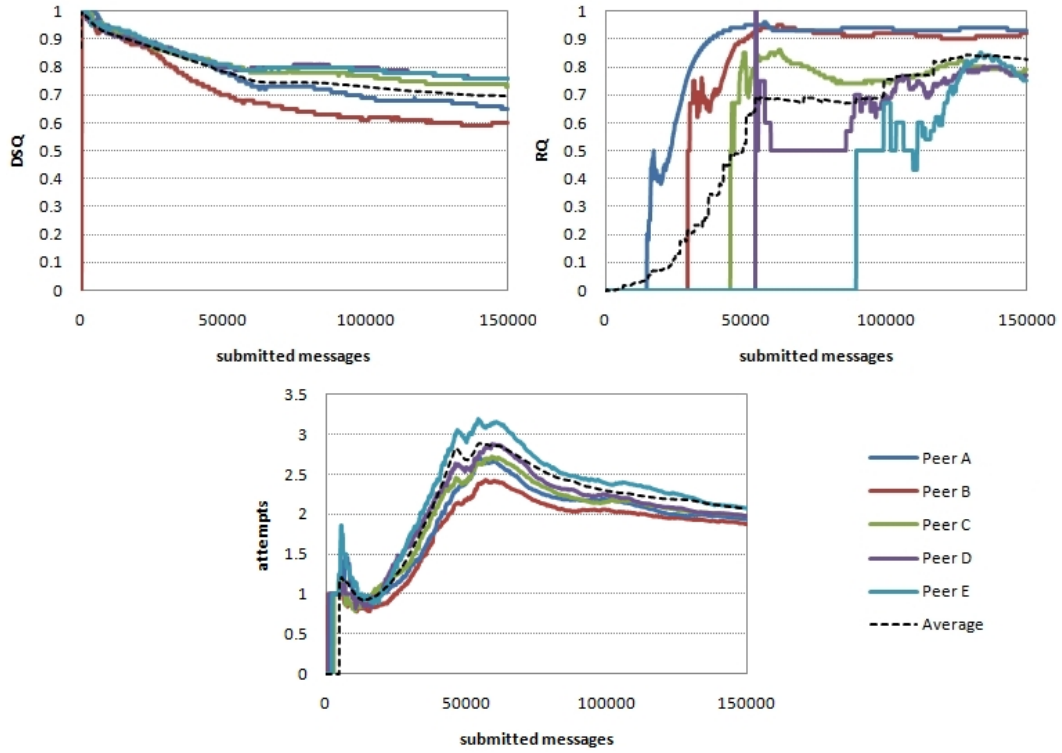


Fig. 2. Evolution of DSQ, RQ and the number of attempts to find the best peer, for five peers randomly chosen among a simulated scenario with 50 peers, having the same query generation rate ($\lambda = 1$) and all privacy criticalities set to 1.

5.2.3 Influence of the query generation rate

In order to see the influence of the query generation rate, we simulated an N-peer game with $N = 5$ using various query generation rates. Like above, each query value was (uniformly) randomly drawn from a set of 1000 possible queries. Figure 5 displays the evolution of DSQ and RQ, as well as the average number of attempts a peer needed to find the best peer, when the five peers had different query generation rates: $\lambda_0 = 0.2$, $\lambda_1 = 0.4$, $\lambda_2 = 0.6$, $\lambda_3 = 0.8$ and $\lambda_4 = 1$. The evolution of DSQ and the number of attempts was similar to the case of five peers with the same query generation rate. Regarding the evolution of RQ, a peer with a comparatively higher query generation rate, that is, a higher λ_i, initially tended to have a comparatively higher RQ until it stabilized around 75%, like the other peers. The rationale is that, in a simulated scenario, a peer generating many queries tends to generate very diverse queries, in such a way that she very soon achieves a homogeneous histogram of submitted queries (high entropy) without need to submit other peers' queries. After a while, those peers generating less queries also attain a homogeneous histogram, and from that moment on, all RQ metrics are similar among peers.

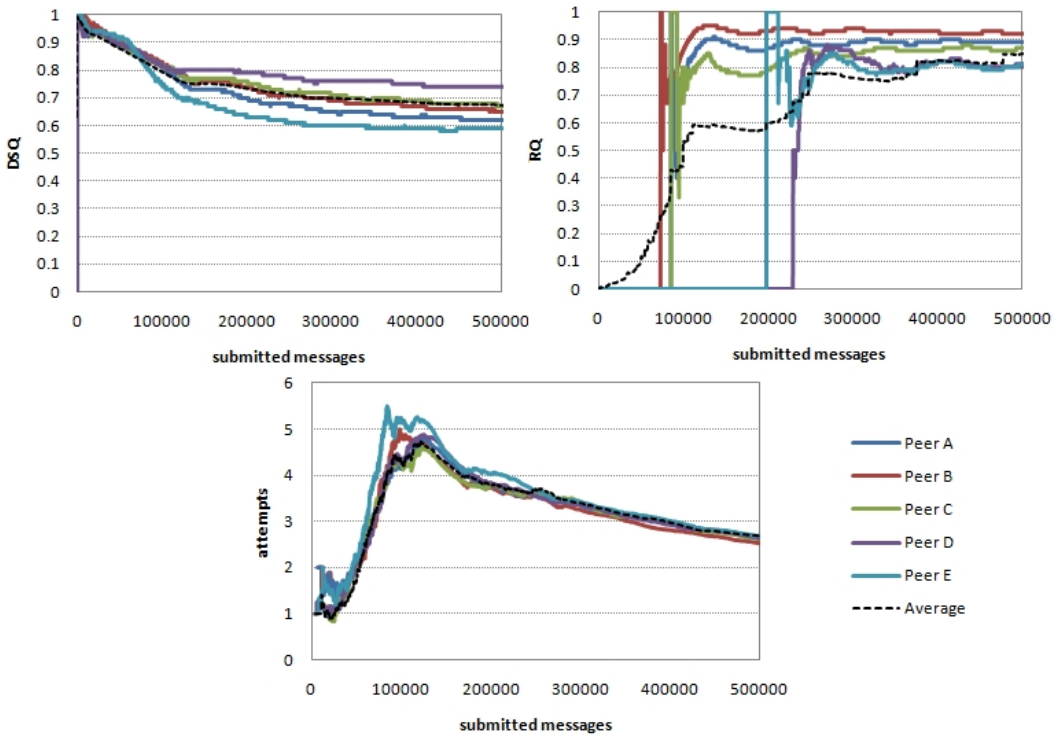


Fig. 3. Evolution of DSQ, RQ and the number of attempts to find the best peer, for five peers randomly chosen among a simulated scenario with 100 peers, having the same query generation rate ($\lambda = 1$) and all privacy criticalities set to 1.

5.3 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 650,000 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 6 shows the evolution of DSQ and RQ, as well as the average number of attempts a peer needed to find the best peer, for those five AOL peers. The graph stops when all peer queries were submitted to DB. The evolution of DSQ and RQ was different for each peer. It depended on the sequence of queries generated by each peer. It can be seen that the average DSQ stabilized around 60%, and the average RQ around 45%. Generally, in this simulation peers were readier to help other peers in submitting their queries to the DB. This is because in a real environment 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. In this case, the average number of attempts to find the best peer was always under 1.

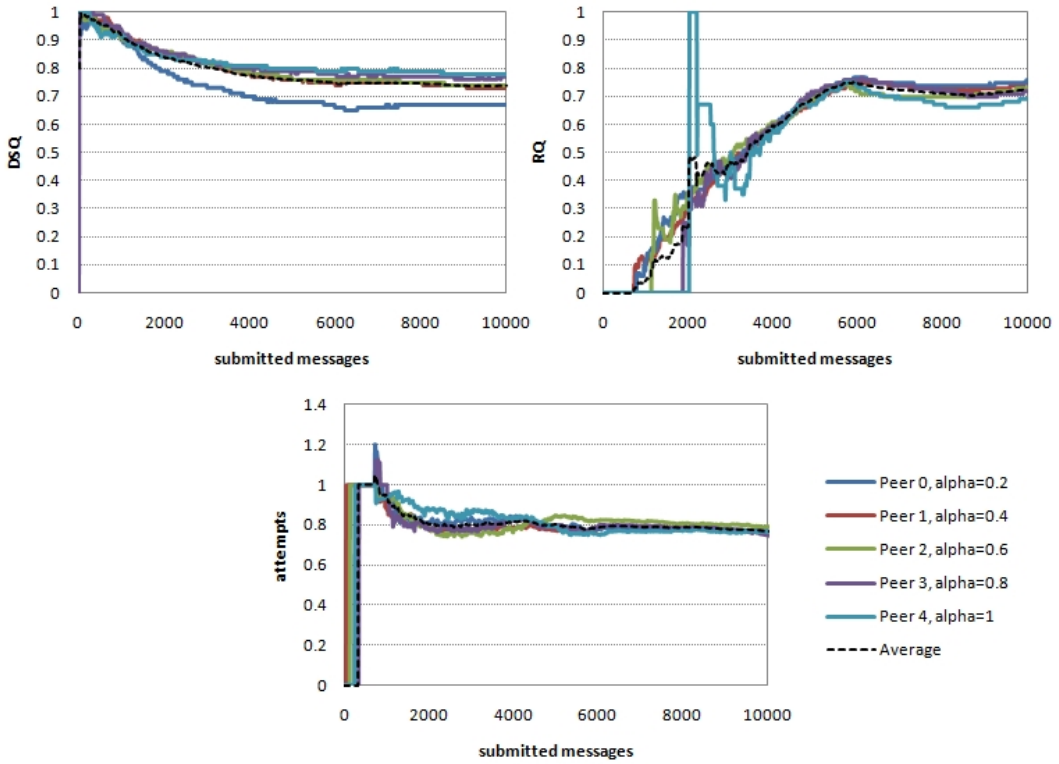


Fig. 4. Evolution of DSQ, RQ and the number of attempts to find the best peer using a simulated scenario with 5 peers having different privacy criticalities and the query generation rate set to 1.

6 Conclusions and future research

We have specified an entropy-based metric for the privacy of peers vs the database and vs other peers in a P2P profile obfuscation system. Based on that metric, we have computed the privacy utility of each peer. When a new query is originated by a peer, the originator and the rest of peers make rational decisions to maximize their privacy. Interestingly enough, those rational decisions often lead to peers helping other peers in submitting queries to the database. Thus, being rationally selfish in terms of privacy leads to helping other peers. In particular, we describe the conditions under which a Nash equilibrium among peers exists; in 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 database.

Empirical results show that the number of attempts needed to find a helping peer stays small regardless of the number of available peers. Consequently, the delay incurred by rational peer-to-peer obfuscation is quite affordable. Also interesting is that, when using real queries, peers are more helpful than with simulated queries; the reason is that real queries are less heterogeneous than

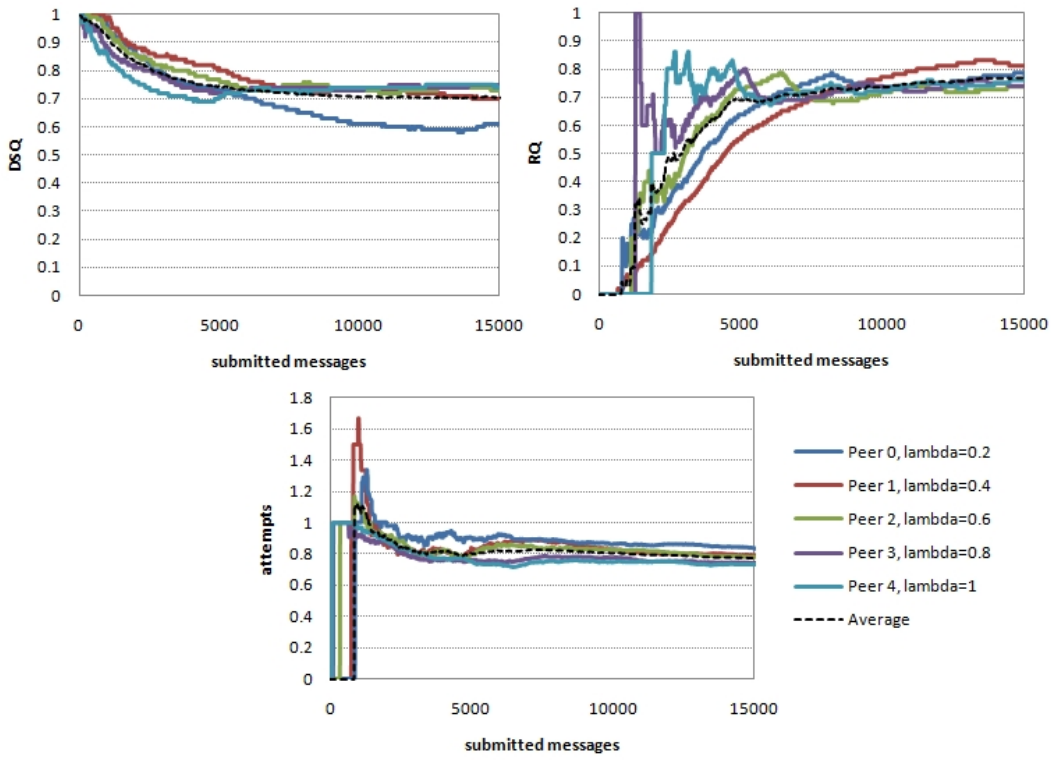


Fig. 5. Evolution of DSQ, RQ and the number of attempts to find the best peer using a simulated scenario with 5 peers having different query generation rates and all privacy criticalities set to 1.

simulated queries, and therefore more queries from other peers are needed to obfuscate one's own queries. Another remarkable empirical result is that, regardless of the number of peers, their query generation rate and their privacy criticalities, all peers end up stabilizing at similar proportions of directly submitted queries and rejected queries, as well as similar number of attempts to find help.

Our proposed model for a rational automated behavior in P2P profile obfuscation systems is meant to be a seminal work that can be extended in a number of ways:

- Generalize the model to multi-hop systems. This is an interesting exercise, even if the slow response time of multi-hop systems may be problematic. Dealing with such systems will involve making rational decisions with incomplete knowledge, as explained in Section 1.1.
- Investigate privacy utilities that take the query semantics into account; to that end, ontologies like WordNet [33] or ODP [24] can be used.
- In addition to privacy, include functionality properties in the peer utilities (for example, query response time, number of hops, etc.).

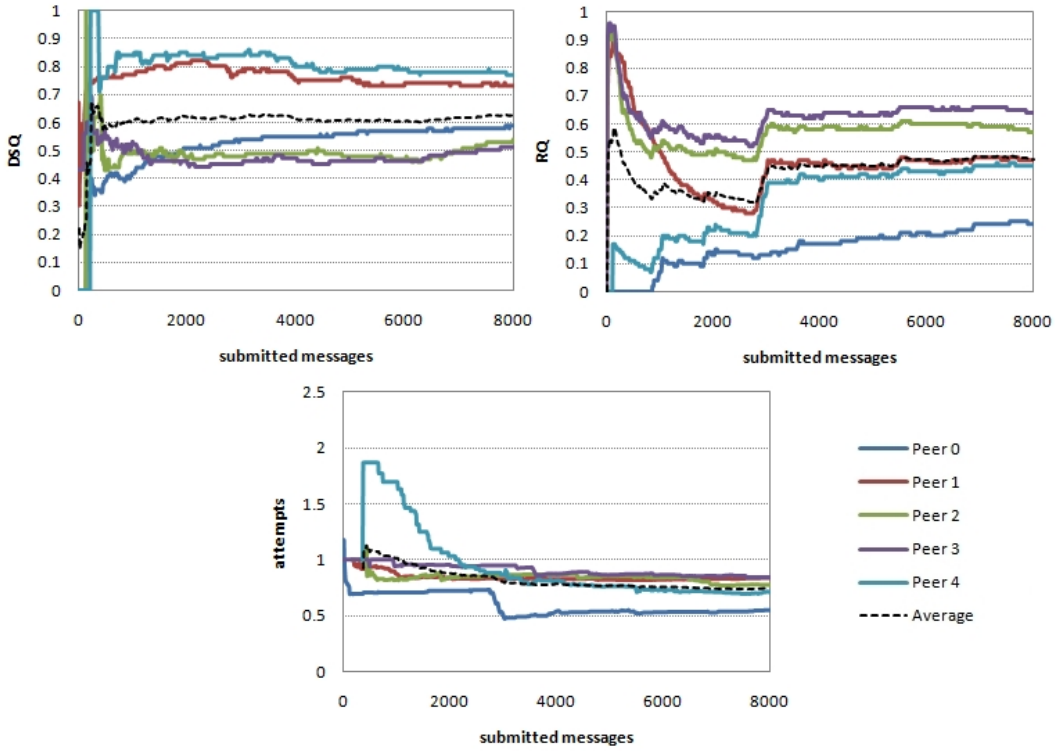


Fig. 6. Evolution of DSQ, RQ and the average number of attempts to find the best peer, for a set of five peers corresponding to five real users in the AOL database (privacy criticalities all set to 1).

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", CONSOLIDER INGENIO 2010 CSD2007-0004 "ARES", TSI-020100-2009-720 "eVerification" 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] D. J. Brenes and D. Gayo-Avello. Stratified analysis of AOL query log. *Information Sciences*, 179(12):1844-1858, 2009.

[5] 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.

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

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

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

[9] J. Domingo-Ferrer. Coprivacy: towards a theory of sustainable privacy. In *Privacy in Statistical Databases-PSD 2010*, LNCS 6344, pp. 258-268, 2010.

[10] J. Domingo-Ferrer and E. Magkos (eds.). *Privacy in Statistical Databases*, LNCS 6344, Springer Verlag, Berlin Heidelberg, 2011.

[11] J. Domingo-Ferrer. Coprivacy: an introduction to the theory and applications of co-operative privacy. *SORT-Statistics and Operations Research Transactions*, vol. 35, special issue: Privacy in statistical databases, pp. 25-40, 2011.

[12] 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.

[13] 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.

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

[15] 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 University Press, Oxford UK, 2009. Software downloadable from: `http://www.mrl.nyu.edu/~dhowe/trackmenot/`

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

[17] iProspect Blended Search Results Study, April 2008, `http://www.iprospect.com/premiumPDFs/researchstudy_apr2008_blendedsearchresults.pdf`

[18] T. Kwon. Privacy preservation with X.509 standard certificates. *Information Sciences*, 181(13):2906-2921, 2011.

[19] 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.

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

[21] Netcraft May 2011 Web Server Survey, `http://news.netcraft.com/archives/2011/05/02/may-2011-web-server-survey.html`

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

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

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

[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*, pp. 393–411, Berlin Heidelberg, 2007.

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

[27] F. Saint-Jean, A. Johnson, D. Boneh and J. Feigenbaum. Private web search. In *Proc. of the 2007 ACM Workshop on Privacy in Electronic Society*, pp. 84–90, 2007.

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

[29] X. Shen, B. Tan and C. X. Zhai. Privacy protection in personalized search. *ACM SIGIR Forum*, 41(1):4-17, 2007.

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

[31] Torbutton 1.2.5. `https://addons.mozilla.org/ca/firefox/addon/2275/`

[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/`