CLIENT-SIDE PRIVACY-ENHANCING TECHNOLOGIES IN WEB SEARCH.

Cristina Romero Tris

Dipòsit Legal: T 1658-2014

Universitat Rovira i Virgili

Department of Computer Engineering and Mathematics

Ph.D. Dissertation

Client-Side Privacy-Enhancing Technologies in Web Search

Author:

Cristina Romero-Tris

Thesis Advisors:

Dr. Alexandre Viejo and Dr. Jordi Castellà-Roca

Dissertation submitted to the Department of Computer Engineering and Mathematics in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computer Science

Cristina Romero-Tris

Client-Side Privacy-Enhancing Technologies in Web Search

Ph.D. Dissertation

Directed by Dr. Alexandre Viejo and Dr. Jordi Castellà-Roca

Department of Computer Engineering and Mathematics

Universitat Rovira i Virgili

Tarragona

2014

[1]To view a copy of this license, please visit: http://creativecommons.org/licenses/by-nc-nd/3.0/

UNIVERSITAT ROVIRA I VIRGILI

DEPARTMENT OF COMPUTER ENGINEERING AND MATHEMATICS

I STATE that the present study, entitled "Client-Side Privacy-Enhancing Technologies in Web Search", presented by Cristina Romero-Tris for the award of the degree of Doctor, has been carried out under my supervision at the Department of Computer Engineering and Mathematics of this university, and that it fulfils all the requirements to be eligible for the European Doctorate Award.

Tarragona, May 15, 2014

Dr. Alexandre Viejo and Dr. Jordi Castellà-Roca, Doctoral Thesis Supervisors

Approved by the University Committee on Graduate Studies:

Acknowledgements

I would like to thank my advisors Alex and Jordi for supporting me during all the process. I also want to thank the CRISES group members, and all the people who have helped me in the elaboration of this thesis. I am also very grateful to Antonio, María Pilar, Rosa, Matías, Clara and the rest of my family and friends for their patience. Finally, I want to thank Joaquín García-Alfaro and all the Telecom-SudParis members for their hospitality during my stay there during the spring 2014.

Abstract

Web search engines (WSEs) are tools that allow users to locate specific information on the Internet. One of the objectives of WSEs is to return the results that best match the interests of each user. For this purpose, WSEs collect and analyze users' search history in order to build profiles. Consequently, a profiled user who submits a certain query will receive the results which are more interesting for her in the first positions.

Although they offer a very useful service, they also represent a threat for their users' privacy. Profiles are built from past queries and other related data that may contain private and personal information. In order to avoid this privacy threat, it is necessary to provide privacy-preserving mechanisms that protect users.

Nowadays, there exist several solutions that intend to provide privacy in this field. One of the goals of this work is to survey the current solutions, analyzing their differences and remarking the advantages and disadvantages of each approach. Then, based on the current state of the art, we present new proposals that protect users' privacy. More specifically, this dissertation proposes three different privacy-preserving multi-party protocols for web search. A multi-party protocol for web search arranges users into groups where they exchange their queries. This serves as an obfuscation method to hide the real queries of each user.

The first multi-party protocol that we propose focuses on reducing the query delay. This is the time that every group member has to wait in order to receive the query results.

The second proposed multi-party protocol improves current literature because it is resilient against internal attacks, outperforming similar proposals in terms of computation and communication.

The third proposal is a P2P protocol, where users are grouped according tho their preferences. This allows to obfuscate users' profiles but conserving their general interests. Consequently, the WSE is able to better rank the results of their queries.

Contents

Chapter 1
Introduction

This chapter introduces the main issues faced in this dissertation. After that, it briefly describes our contributions to the field. Finally, the structure and organization of this thesis are defined.

Contents

We start with the motivation in §1.1, followed by the main contributions performed in §1.2, and the organization of this PhD dissertation in §1.3.

1.1 Motivation

The volume of information available on the Internet grows every day. The number of web pages is estimated to nearly double every three-year period, according to the study presented in [Netc 12]. For example, 620,132,319 websites were counted in September 2012, a 27.81% more than in September 2011.

For this reason, in recent years, the problem of acurately retrieve information from the Internet has received a lot of attention. Web search engines (WSEs) are tools which allow users to find specific information through the use of keywords. When a user submits a query, the WSE searches for the required information among the billions of indexed web pages, and returns the search results in the form of ranked documents.

During this process, the WSE automatically records the submitted query and some related information, which are often called *query logs*. For example, Google's Privacy Center [Goog 13] states that their *query logs* include the user's query, IP address, browser type, browser language, date and time of the request and a reference to one or more cookies that may uniquely identify the user's browser.

Once recorded, *query logs* are processed and analyzed in order to build *user profiles* that represent distinctive features about users' search behaviour and information needs. Then, *user profiles* are employed for different purposes. One of them is called *personalized search*. This technique improves the WSE's performance by presenting the search results ranked according to the user's interests. As stated in [iPro 08], 68% of WSE users click a search result within the first page of results, and 92% of them click a result within the first three pages of search results. Consequently, in order to be successful, WSEs try to show the links which are more relevant for a particular user in the first result pages.

Nevertheless, many complications may arise when discerning users' interests. An example of this happens when looking up the word "Mercury". This term can refer to the planet Mercury or to an element in the periodic table. The concept *disambiguation* is the process of identifying the correct sense when a certain word has different senses. In the WSE scenario, the query disambiguation process requires the WSE to know the user's interests and the query context, which can be obtained from the *user profile* [Daou 09].

Although WSEs play an important role in the use of the Internet, they can also raise concerns regarding the privacy of the users. The different logs stored by a WSE contain sensitive data that can be combined to disclose information of a certain individual. The Universal Declaration of Human Rights [Univ 48], article 12, states that "*No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation*". It also recognises the right of everyone of being protected against such interferences or attacks.

The current practice of most WSEs that log and analyze queries, represents a threat for the privacy of the users. For example, selling profiles to third parties (*e.g.,* advertisers, media, etc.) significantly increases the economical benefits of the WSE [E St 10]. This kind of activity is a threat for users' privacy because the disclosed data may contain Personally Identifiable Information (PII). In the *Guide to Protecting the Confidentiality of Personally Identifiable Information* [Nati 10], PII is defined as any information about an individual maintained by an agency, including:

1. Any information that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records.

2. Any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information.

When *user profiles* contain PII, and they are disclosed to third parties, users' privacy is seriously threatened. The survey conducted in [Anto 10] establishes a baseline of Internet users' privacy concerns. The survey results show that the respondents' top concern is information transfer, including disclosure of their patterns and the trading/selling of PII to third parties.

Although this information is not generally directly linked to the user's name, it is still possible to pseudo-identify a certain individual:

- The IP address of the computer which is used to submit the queries pseudo-identifies the owner. Nevertheless, the use of *dynamic* IP addresses makes identification difficult.
- Browser cookies identify uniquely the browser of the user [Ye 09].
- Browser search bars assign a unique ID to the browser of the user. This ID can be used for identification purposes.
- Another kind of pseudo-identifier called *browser fingerprinting* is studied in [Ecke 10]. In this work, it is shown that browsers reveal much information about the version and configuration information to websites, which can be used as a fingerprint to track an individual. Specifically, they propose an algorithm that unequivocally identifies the browser of a certain user with a probability of 99.1%.

These pseudo-identifiers represent a unique user and they are linked to all the information gathered from her. This situation poses a serious threat to the users because their associated pseudo-identifiers may reveal their real identity in several scenarios:

1. An *Internet Service Provider* (ISP) can connect the IP address linked to a bunch of queries with the complete name of the user who submitted them.
2. Let us assume a user who logs in an account associated to the WSE and submits queries (e.g., a Google user logged in Gmail). The WSE is able to connect these queries with that account. This account probably contains the real name of the user who owns it.

3. Let us imagine a user who submits a query about personal information which identifies him uniquely: her name, national ID, etc. This kind of queries are called *vanity queries* [Jone 08], and they easily link the user's pseudo-identifier with her real identity.

4. Even though a single query might not reveal the real identity of a certain user, the aggregation of several queries might cause this situation.

In practice, points three and four (queries about personal information and aggregation of queries) have been shown to be effective in identifying users. An example of this is the AOL scandal, where 20 million queries made by 658,000 users were publicly disclosed [M Ba 05]: Thelma Arnold, user of the AOL's WSE, was identified by her searches submitted over a three-month period. All these queries were hidden behind a pseudonym (in this case it was the number 4417749). This number was assigned to protect the searchers' anonymity. However, the aggregation of hundreds of queries, some of them containing personal information (*e.g.*, her own name, the city where she lives, etc.), was enough to identify and profile her.

The AOL scandal is a practical example of the risks of unprotected web search and query log disclosure, but it is not the only one. According to [Coop 08], there are four categories of *privacy risks* caused by query logging:

1. *Accidental or malicious disclosure*

   Disclosure of data that contains private information is an obvious risk of query logging. Even for a WSE that does not intentionally disclose query logs, there exists a risk of accidental disclosure. This might be the result of security flaws or mistakes related to a purposeful controlled disclosure. For example, in 2006 AOL released a file with twenty million searches generated by its users [M Ba 05]. This incident had serious consequences since personally identifiable information was present in many of the queries.

   Individuals also face threats from malicious disclosure. In such cases, an attacker or a dishonest employee or researcher purposely discloses data that was meant to be kept private or that may cause harm to others in some way.

2. *Compelled disclosure to third parties*

   Query logs may be subject to a subpoena as a part of civil litigation between individuals or organizations. For example, a seach engine company may be compelled to disclose queries related to an individual involved in a legal case as part of the evidence provided to the court. In these situations, search engines risk their reliability in front of their users if they comply. If they do not comply, they may face potentially long and costly litigations by the requesting entity.

This kind of privacy risk attracted a lot of attention in 2006, when the U.S. Department of Justice issued subpoenas to AOL, Google, Microsoft and Yahoo as part of its litigation of an Internet child safety law [K Ha 06]. The Department of Justice requested several months of query logs to use as evidence that Internet filters were not adequately protecting children from adult content. This requested was largely viewed as massively overbroad and not extrictly necessary to the case [M Ra 06]. Google refused to comply with the subpoena and submitted a smaller set of information than what the requested intended. However, the other search engines agreed to disclose what the Department of Justice was originally requesting.

3. *Disclosure to the government*

   The arrival of the Internet entailed a great supply of stored data that may be of interest to the government. Government authorities often have valid, compelling and urgent needs to examine query logs. However, the deliver of this sort of information should be carefully considered in order to avoid surveillance that is overbroad, unjustified or erroneous.

   Regrettably, countries such as the U.S.A or other countries in Europe. have laws that have not been updated with technological advances. Therefore, these laws are sometimes ambiguous, allowing the government to access the information under weak pretexts.

4. *Misuse of user profiles*

   The retention of query logs may allow the creation of detailed profiles about the interests, preferences and behaviors of the users. These profiles may be particularly appealing for marketing purposes, both internal to the search engine (e.g., sponsored links), and as data sets provided to third parties. They may also be used as a tool to calibrate price discrimination, where a marketer charges each consumer the maximum price that she is determined to pay for the same item.

In addition to these risks, [Coop 08] also remarks the potential for logs to be erroneously linked to the wrong individuals (e.g., linking the logs from a computer employed by multiple users to the same person). Similarly to identity fraud, associating queries about diseases, sexual orientation, politics, etc. to someone who did not generate them may have serious negative consequences for the affected user.

1.2 Contributions

As a response to the privacy concerns, there are some *Privacy Enhancing Technologies (PETs)* in the literature whose objective is to protect users' privacy in front of WSEs. The solutions can be addressed from two different points:

- **Server-side**. The WSE wants to share or outsource the collected query logs without putting users' privacy at risk. There are several methodologies in the literature that study how to anonymize these logs, such as Privacy Preserving Data mining (PPD) or Statistical Disclosure Control (SDC).
- **Client-side**. This type of mechanisms assume that WSEs have no interest in protecting users' privacy or do not trust WSEs to correctly protect users' privacy. Techniques inside this category allow users to obfuscate the information that WSEs know about them.

This dissertation only focuses on the second point, the client-side techniques. We assume that once the personal data are gathered, users can do nothing to prevent WSEs from putting their privacy at risk. Therefore, we study the mechanisms that users can employ to protect their privacy before the personal data is gathered and analyzed.

One of the objectives of this document is to present a survey on the current client-side technologies that allow private web search. This includes the classification and analysis of advantages and disadvantages of the different kinds of approaches.

This work also contributes to private web search by offering new proposals. Based on the study of current literature, we identify some points of improvement and present three new schemes:

- The first proposal analyzes an existing protocol, the UUP protocol [Cast 09], and modifies it in order to obtain a lower time of response. The resulting protocol obtains the lowest query delay in the literature of multi-party private web search.

- The second proposal is also based on the UUP protocol, but in this case it improves the *level of security* . The new protocol resist the presence of users who do not behave properly (i.e., adversaries), and offers a shorter response time than similar proposals in the literature with the same level of security.
- The third proposal is a Peer-to-Peer (P2P) protocol that groups users according to their preferences. Inside each group, users exchange their queries before submitting them to the WSE.

1.3 Structure of this document

Chapter 2 surveys and classifies the different existing proposals that provide privacy in WSEs. Each proposal is described and its main advantages and disadvantages are discussed.

Chapter 3 gives some background about cryptographic techniques employed in subsequent chapters. Our contributions to the field are described in Chapter 4, Chapter 5 and Chapter 6.

Finally, Chapter 7 includes the conclusions of the work. In addition, the main lines of future research are described.

Chapter 2
State of the Art

The work contained in this chapter analyzes the different mechanisms that users can employ in order to protect their privacy in front of WSEs. It includes a novel approach to classify current client-side proposals, as well as a study on how they work, their advantages and their disadvantages. Furthermore, we propose a common evaluation framework composed by eight properties (extracted from the reviewed literature) that affect users' experience when protecting their privacy from the WSE. We then compare all the reviewed systems according to this common evaluation framework in order to determine which proposals enhance users' experience.

Contents

Section 2.1 proposes a model for classifying the different privacy enhancing technologies applied to web search that can be found in the literature, and analyzes every proposal inside each category, studying their advantages and disadvantages. In Section 2.2 we compare these technologies and discuss some properties that affect users' experience. Finally, Section 6.6 provides some concluding remarks about the state of the art.

2.1 Client-side Privacy Enhancing Technologies for Web Search

This section surveys the current Privacy Enhancing Technologies (PET) that allow private web search from the client-side perspective. This includes the classification and analysis of advantages and disadvantages of the different kinds of approaches.

According to [Oliv 03] and [Oliv 04], PETs can be classified into four categories *(Private Communication, Anonymity, Personal Control, and Organizational Safeguards)*. In the work presented in [Bran 11], this classification is applied to privacy-preserving web search schemes:

- *Private Communication*. This category consists of technologies that allow a user to communicate content only to the specified recipient(s), regardless of who is listening.
- *Anonymity*. Pfitzmann and Koehntopp [Pfit 01] define anonymity as "the state of being not identifiable within a set of subjects". The anonyimity schemes are mainly based on Chaum's mix networks [Chau 81] or based on the notion of a proxy. Consequently, these schemes often require the colaboration of an external entity.
- *Personal Control*. The idea of the schemes inside this category is to allow users to ensure that their personal information is only used in a manner that corresponds to their privacy policies. These privacy policies can be defined and modified by each user.
- *Organizational Safeguards*. This category is similar to Personal Control, although it focuses on the organization side, and not the client side. The Organizational Safeguards refer to the use of technology to ensure that the organization complies with its privacy policy as well as with the preferences of the users.

These four categories comprise all the existing methods to protect users' privacy in front of WSE. However, for our purposes, this classification covers a too wide range of proposals (e.g., *organizational safeguards* only apply to the server side, which is out of the scope of this dissertation). For this reason, we have extracted some ideas from the work done in [Oliv 03] and [Oliv 04] and we have constructed a new classification that is specifically designed for client-side approaches.

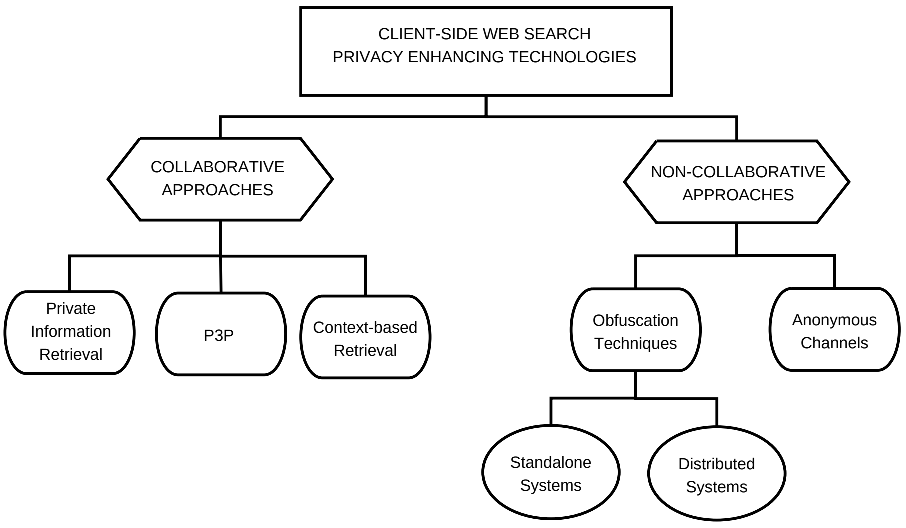


Figure 2.1: Classification of Web Search Privacy Enhancing Technologies that work on the client side

Figure 2.1 shows our proposal for classifying client-side technologies that protect users' privacy in front of WSE. At the highest level, we classify approaches according to their needs of collaboration with the WSE. On one hand, collaborative approaches require a certain level of cooperation from the WSE. Note that these techniques are not considered server-side because users *actively* participate in the process. Furthermore, an important characteristic of collaborative approaches is that when the WSE does not cooperate, the user immediately detects it. This does not happen with server-side approaches, since they are asynchronous and transparent to the user. On the other hand, non-collaborative approaches protect users' privacy without any help from the WSE, and do not require any change in the server side (the WSE). Regarding the original classification from [Oliv 03] and [Oliv 04], *personal control* and *private communication* would be included inside collaborative approaches, while non-collaborative approaches would correspond to *anonymity*, divided into sub-categories *anonymous channels* and *obfuscation techniques*. Next, we describe the categories and subcategories from Figure 2.1 in a higher detail, as well as the main schemes in the literature that fall into each of them.

2.1.1 Collaborative Approaches

As stated above, in collaborative approaches users and WSEs work together in order to protect users' privacy. Inside this category, three other subcategories can be found: Private Information Retrieval, P3P, and Context-based Retrieval. Each of these subcategories is subsequently described.

Private Information Retrieval

Private Information Retrieval (PIR) schemes allow a user to retrieve information from a database privately, i.e., without the server learning what information was retrieved. With a PIR scheme a user can search the documents stored in the database, and thus recover the documents of interest on their own. Therefore, the problem of submitting a query to a WSE while preserving the user's privacy can be seen as a PIR problem.

The first PIR protocol was presented in [Chor 97] and [Chor 98]. These scheme is based on several servers which hold the same database and cannot communicate between them. This is not applicable to our study, since in the WSE scenario there is only one server. Even in the case of a WSE with several servers, it is not realistic to assume that servers cannot communicate between them.

A more appropriate scheme for the WSE scenario is single-database PIR. The first reference was presented in [Kush 97] (see [Ostr 07] for a detailed survey on single-database PIR protocols). This schemes are specific for scenarios with only one server that holds the database. However, according to [Cast 09], in practice they suffer from some fundamental problems that still make them unsuitable for WSEs:

1. PIR schemes are not suited for large databases. In the PIR literature, the database is usually modeled as a vector. The user wishes to retrieve the value of the i-th component of the vector while keeping the index i hidden from the server which holds the database.

   Let us assume that the database contains n items. A PIR protocol will try to guarantee maximum server uncertainty on the index i of the record retrieved by the user. This is done by accessing all records in the database. Note that if a certain user only accesses a part, it will be easier for the server to know the real interests of this user. The cost of accessing all records represents a computational complexity of $O(n)$.

2. When accessing the records in the database, it is assumed that the user knows their physical location. In WSE, this situation is not realistic because the database is not managed by the user. In [Chor 97], the authors propose the use of a mechanism which maps individual keywords to physical addresses. According to that, the user can submit a query consisting on a keyword and no modification in the structure of the database is needed. However, this model cannot be applied to our scenario, since WSE do not map keywords to physical addresses.

Platform for Privacy Preferences (P3P)

Another methodology to protect privacy with the collaboration of WSEs is the Platform for Privacy Preferences (P3P). It was created by the World Wide Web Consortium (W3C) to make it easier for website visitors to obtain information about the privacy policies of the sites that they visit. It is a framework that allows users to automate the protection of their privacy by expressing their privacy preferences. When a user encounters a website that does not conform to these preferences, a special software alerts the user or takes other actions such as blocking cookies.

The work presented in [Cran 06] proposes an application of P3P to WSEs. The basic idea is that when a search term is entered, the WSE retrieves the P3P policies for all the query results. Then it compares them to the privacy preferences that the user has previously defined. According to this comparison, the WSE re-ranks the results, and those matching the user's preferences are presented first. This method is analyzed in [Tsai 09], using 15,000 search queries from 460 participants over a 10-month period. Results show that by displaying privacy information together with search results, users are more likely to visit websites that provide privacy indicators and, among them, those who have higher levels of privacy.

Related to P3P, a policy-based system called Do-Not-Track was originally proposed in 2009 by researchers Christopher Soghoian, Sid Stamm, and Dan Kaminsky, as explained in [C So 12]. It has been later studied in works like [Maye 11], [Beck 12], and [Tene 12], and it is currently being standardized by the World Wide Web Consortium (W3C). The Do-Not-Track (DNT) is a HTTP header field that requests the web application to not track an individual user. Similarly to P3P, it relies on the web application compliance in order to be effective.

This kind of schemes has been widely criticized in the literature. For example, [Hoch 02] suggests that industry efforts for self-regulation prevent the U.S. from passing a comprehensive privacy law, leaving users in a weaker alternative. In [Reay 09], the authors collect all available P3P documents from the 100,000 most popular Web sites and analyze their adherence to legal mandates. Results show that websites do not generally even claim to follow all the privacy-protection mandates in their legal jurisdiction, including European Union nations, Canada, Australia, and websites in the USA Safe Harbor program. Additionally, there is no mechanism by which users can verify that the website complies its own posted privacy policy. Other critics [Elec 00] claim that P3P is hard to implement, lacks enforcement provisions, and will never have enough adopters.

On the other hand, works like [Cran 12] suggest that even if these kinds of mechanisms are not enough to protect privacy, they are complementary to privacy regulation. Once users are sure that their information is protected at least at a baseline level, P3P policies have the potential to provide meaningful control over secondary data uses and sharing. However, [Cran 12] also remarks that some other enforcement mechanisms are needed to ensure that users' choices are respected.

Context-based Retrieval

As previously justified, personalized search improves the quality of service provided by the WSE by decreasing search ambiguity and returning results that are more likely to match a particular user's interests. However, allowing WSEs to store query logs on its servers has many already described privacy issues. Consequently, there are some alternatives in the literature that suggest to store the search history or the user profile on the client's machine.

For example, the User-Centered Adaptive Information Retrieval (UCAIR) project [Shen 05] aims at developping a new kind of WSE capable of optimizing the search results according to each individual's interests. In order to do this, they collect and exploit available user context from submitted queries and clicked results. More specifically, the proposed system has three connected modules with different functions:

1. The user modeling module, which captures user's search context and history information, including the submitted queries and any clicked search results.
2. The query modification module, which selectively improves the query formulation according to the interests represented in the previous module.

3. The result re-ranking module, which re-ranks search results before they are presented to the user.

The combination of these three modules allows to represent users' interests and re-rank search results according to these interests. This alternative is considered inside the collaborative approach category since WSE and users participate together during the search process in order to obtain the final results: the WSE receives the query and returns the results, and these are re-ranked in the client-side.

Based on a similar idea, the protocol presented in [Xu 07] allows the user to choose the content and degree of detail of the profile information that is exposed to the WSE. The protocol is responsible for building the profile of the user in the client side and then, the user determines the content from this profile that will be revealed to the WSE when a query is submitted. In this case, the collaboration between the WSE and the user is needed in the sense that the WSE must be prepared to receive a profile together with the query, and personalize the results according to that profile and not according to the profile stored in the server side.

In order to build the profile sent to the WSE, the system has a user interface where the user selects three different parameters:

1. The personal data sources that can be employed to build her profile (e.g., browser history, emails, documents, etc.).

2. A threshold parameter (*minDetail*) that indicates the level of detail that will be revealed to the WSE.

3. A parameter α that represents the weight assigned to the relationship between the user profile and the query results (i.e., the extent to which the results returned by the WSE should consider or ignore the user profile).

The most challenging aspect of these methods is how to infer the user profile based on the user's ongoing behavior, and how to represent it accurately. Recent studies like [Leun 12] [Kram 13] build on the same idea of using context to personalize results. These works adjust some parameters of the stored profiles, such as conceptual relationships and concept features. The idea is the same, but they are able to control the exposed user information more accurately and maintain better ranking quality than [Shen 05] or [Xu 07].

Nevertheless, there are still some disadvantages of this kind of alternatives:

- Performing the re-ranking at the client side may not be as effective as doing it at the server side. As pointed out in [Ucai 05], the client has no global index for all websites and hence, the general retrieval function cannot be probably controlled at the client side.

- Sending a part of the profile to the WSE together with the query may allow the WSE to build a correct user profile after several executions. Moreover, in order to decide which part of the profile is sent to the WSE, the user is frequently required to adjust some parameters. This means that the user has to actively participate in the process and have some expert knowledge about it.

2.1.2 Non-collaborative Approaches

Collaborative Approaches assume that the WSE cooperate with the user in executing the protocol. Nevertheless, in some scenarios this assumption is not realistic because the WSE has no motivation to protect the privacy of the users and limit its profiling ability. Accordingly, there are other kinds of techniques, named non-collaborative approaches, which allow users to be the only responsible for their privacy. These schemes do not expect any collaboration from the WSE.

Non-collaborative approaches can be further classified into two subcategories: *obfuscation techniques* and *anonymous channels*. The former is based on the generation of enough noise to distort the user profile that the WSE stores. The latter focuses on creating an anonymous channel between the user and the WSE. Next, both subcategories are detailed.

Obfuscation Techniques

Obfuscation techniques are based on introducing additional random "cover" traffic, intended to obscure the true *communication pattern* between sender and receiver. The property that actual messages sent by a subject are indistinguishable from random noise is known as *unobservability*.

There are several works based on this solution that have been implemented in the literature. These works can be further classified regarding the number of users that participate in the protocol between *standalone* and *distributed*. The former approach allows that one user alone protects her privacy in front of the WSE. The latter requires that a group of users work together in order to protect the privacy of each member of the group.

Standalone Systems Standalone schemes are based on generating a stream of synthetic queries that are used to hide the real queries of the user. Synthetic queries are submitted together with the real queries, obfuscating the profile that the WSE owns of an individual. There are two different approaches in standalone systems, depending on the way that synthetic queries are generated. If the synthetic queries are somehow semantically related to the real queries, the obfuscated profile will still be usable and the WSE will be able to personalize the results. If the synthetic queries are semantically unrelated to the real queries, the profile will be heterogenous and the personalization will be less accurate. Note that this does not mean that one altenative is better than the other, since in the second case the WSE will know less details about the user's interests, which increases her level of privacy. Different users may have different perceptions of the value of their own privacy and hence, they may be interested in different kinds of approaches. Next, some of the most renowned standalone systems in the area are classified according to the sematical relation between the synthetic queries and the real queries generated by the user.

1. *Synthetic queries semantically unrelated to real queries.*

   The most notable example of these systems is TrackMeNot (TMN) [Trac 13]. TMN is a Firefox plugin designed to achieve privacy in web search by obfuscating the queries of the users. This is done by using a stream of machine-generated queries that act as a decoy and which hide the real queries. The idea is that providing enough "noise" around the true search patterns of the user, will make it very difficult to distinguish the queries made by TMN from those actually made by humans.

   The synthetic queries are picked from a variety of sources that provide terms. At the beginning, an initial "seed list" is generated using RSS feeds. During the executions, the queries are pulled from this list and submitted to the WSE. The list is dynamic, so it is updated with new RSS feeds periodically.

   The user can customize many parameters that affect TrackMeNot behavior. For example, she can select the WSE (e.g. AOL, Bing, Yahoo, Google, etc.) to which the queries will be submitted. She can also specify the frequency of the queries submission, as well as choosing the website with RSS feeds from which the seeds are picked.

One of the problems of TrackMeNot is that it sumbits random queries to the WSEs when the activity of the users is low. This is done to prevent the system from affecting the normal work of the users. Nevertheless, this can be a threat for users' privacy: for each user, the WSE is able to divide all her queries depending on whether or not they have been submitted during working hours (according to the time-zone of the user). Probably, all the queries submitted out of the working hours have been sent by TrackMeNot. The period of time between two different queries can also be used to the same purpose: it can be assumed that when the users are working, they do not submit only one query but several in a short period of time. This gap between queries can be used to deduce whether or not a certain query has been submitted by TrackMeNot.

Additionally, TrackMeNot queries are semantically unrelated in most cases. This means that while users normally submit sequential queries on the same subject, TrackMeNot queries have different subjects randomly chosen. The WSE can use the timing and semantic features to distinguish and detect TrackMeNot queries. In fact, works like [Chow 09], [Pedd 10], and [Al R 12] show that it is possible to distinguish real queries from synthetic queries. These works are based on the idea that machine-generated queries do not have the same features as human-generated queries. For example, [Pedd 10] develops a classifier which is very accurate in identifying TrackMeNot queries. The authors use a dataset of 3 months of user queries from the available AOL search logs. Then, they apply two different categories of machine learning algorithms:

(a) Clustering algorithms: These algorithms group the elements of the dataset into clusters, where the elements of the same group share some common features. The authors tested the performance of three clustering algorithms: SimpleKMeans, Farthest First and EMClusterer.

(b) Classification algorithms: These algorithms are based on labels. They need a learning phase made over a training set of queries. In this training set, each element has its own label. After the learning phase, the algorithm is able to label the elements on a new test set.

In this case, the authors test the performance of six classification algorithms: Logistic (Regression), Alternating Decision Trees (ADTree), Naive Bayes, Random Forest, Random Tree and ZeroR.

The results obtained using these algorithms were very accurate in identifying the TrackMeNot queries. In fact, the mean of misclassification rate was only 0.02%.

GooPIR [Domi 09b] is another application based on a query obfuscation protocol. It submits fake *words* to the WSE together with the authentic query. This system is based on a concept called "h(k)-private information retrieval", where k is a non-negative integer representing the number of queries that a user submits (one real query and $k-1$ synthetic queries), and $h(\cdot)$ is a function such that $h(k) \geq 0$. Their protocol provides h(k)-private information retrieval because any intruder views the query of a user, denoted as q_0, as a random variable (Q_0) whose Shannon's entropy is $h(Q_0) \geq h(k)$. Informally, this means that the intruder can see the query as a variable (Q_0) that can take several values. Among them, the intruder cannot unequivocally determine which was the exact query (q_0) submitted by the user.

GooPIR uses a Thesaurus to obtain the *words* which are mixed with the real queries. Thesaurus provides the keywords along with their relative frequencies. Keyword frequencies can be obtained based on the appearances in a text collection (e.g. newspapers) or from the frequency output when the keyword is looked up in a WSE. These frequencies are used to determine the words that can be submitted together with the real query. Keywords with similar frequency to the query frequency provide a higher entropy.

Unlike TrackMeNot, there is no current proposal that proves that GooPIR queries can be differentiated from real queries. However, [Bals 12] points out that GooPIR does not provide perfect query indistinguishability when various sets of queries are taken into account. For example, let us consider $k = 3$ and a user who generates the query "pizza". Then, GooPIR will submit it together with two other synthetic queries (e.g., "mouse", "pizza", "plastic"). Then, let us consider that consecutively afterwards ("spaghetti", "muscles", "school") and ("hockey", "risotto", "purple") are submitted. Then, an attacker (e.g., the WSE) can find a correlation in the combination of terms and remark the user's interest in italian cuisine, assigning to her profile the queries "pizza", "spaghetti" and "risotto", and discarding the rest.

Moreover, another important drawback of GooPIR is that it uses a Thesaurus in order to decide which words can be added to the search. According to that, GooPIR can only submit words. Full sentences are not addressed (note that sentences cannot be formed by random words).

Plausibly Deniable Search (PDS) [Muru 09] is similar to GooPIR in the sense that $k-1$ fake queries are submitted together with the real query. One of the differences is that PDS subsitutes the real query by a *canonical query* (i.e., the real query without grammar errors or typos). This prevents the WSE to identify the real query by the human-writing mistakes that it may include. Then, the system uses a Latent Semantic Indexing-based approach [Deer 90] to generate the $k-1$ synthetic queries in such a way that the final set of k queries relate to different topics. Consequently, all the k queries have the same probability of being the query generated by the user (i.e., they are equally plausible).

In order to prevent the correlation attack appliable to GooPIR, PDS ensures that two semantically related queries are obfuscated by synthetic queries that are also semantically related. As an example of this, let us consider that the query "pizza" mapped to "italian cuisine" was obfuscated with queries about "pop songs" and "computer networks". Then, the next query about "italian cuisine" (e.g., spaghetti) will also be obfuscated with synthetic queries from the "pop songs" and "computer network" categories, preventing an attacker to identify a unique common subject in the submitted queries. However, as described in [Bals 12], the problem of this system is that it assumes that the adversary will map the queries into the same categories as PDS. For example, let us consider the query "baby food" that PDS maps into "nutrition", and the query "gynaecologist" mapped into "health". Since the queries do not belong to the same category, the synthetic queries will not be semantically related. However, an adversary that maps both queries to the category "pregnancy" will still be able to find correlations in the submitted queries and hence, discard the synthetic queries.

2. *Synthetic queries semantically related to real queries.*

   The general idea under this proposal is again the use of synthetic queries that are submitted together with the real queries of the user. The novelty of the approach regarding TrackMeNot, GooPIR or PDS, is that it considers similarities between the topics of synthetic and real queries.

For example, the scheme presented in [Viej 12a] uses the Open Directory Project (ODP) [ODP 13] hierarchy to infer the real interests of the user and limit the semantic distance between fake and real queries. In order to do this, every time that a user wants to submit a real query q, the following steps are executed:

- The system searches the real category C_r in the ODP hierarchy that includes q.
- It retrieves some fake categories $C_f^1, C_f^2, \ldots, C_f^m$ related to C_r, e.g., more specific or more general categories, another child of the same parent category, etc.
- Some terms from each fake category C_f^i are picked and submitted to the WSE together with q.

The performance of the proposed scheme is evaluated using real queries extracted from the AOL's dataset [M Ba 05]. Results show that WSEs cannot distinguish synthetic queries from real queries, and that although the profiles of the users are obfuscated, the search results that the WSE returns are similar to the results that the user obtains with a non-obfuscated profile. Regarding the performance, the synthetic query generation time is less than 700 ms.

The problem of this scheme is that it does not address queries with multiple words. If a query is formed by several words or terms, they are treated individually and their categories are separately retrieved from the ODP hierarchy. For example, the query "white pages" would be splitted into "white" and "pages", and the corresponding ODP categories would not reflect the real meaning of the query.

In order to solve this problem, the work presented in [Sanc 13] applies a linguistic analysis preprocessing to the query before the ODP categories are retrieved. The use of Natural Language Processing (NLP) techniques (i.e., Sentence Detection, Tokenization, Part-of-speech tagging, etc.) allows to interpret *complex queries* and find the proper category in the ODP hierarchy.

Accordingly, the synthetic queries are more accurately generated and the semantics are better preserved than in [Viej 12a].

A slightly different idea is presented in [Aram 12], where the authors design a system called QueryScrambler. The objective of the QueryScrambler scheme is to retrieve the web results related to a query from the WSE, without revealing the query and the actual interests of the users. Therefore, the main difference regarding [Viej 12a] and [Sanc 13] is that the real query is never submitted. Instead, given a query, a set of scrambled queries that correspond approximately to the same interest are generated. These scrambled queries are submitted to the WSE who returns the results. Then, these results are reconstructed by a process called descrambling, which attempts to produce a ranking similar to the one that the real query would have obtained. In order to obtain the set of scrambled queries, the method employs an ontology called WordNet. For each query term, a set of related terms is generated following again the hypernymy and hyponymy relations in the ontology.

In order to perform the descrambling, two methods are proposed: fusion and local re-indexing. The fusion method considers the number of times that an item appears in the results that belong to the scrambled queries. The local re-indexing consists in submitting the real query to a local engine that returns results only appearing in the union of scrambled results.

The method is evaluated by running an offline experiment. The results indicate that the QueryScrambler gets up to 25% of the top-50 target results. The disadvantage of the QueryScrambler is that it does not protect the user from adversaries that know the method. These adversaries could potentially reverse the procedure in order to get the real interests. As a minor disadvantage, the method is not able to generate scrambled queries from adverbs or adjectives, since WordNet does not accept them. Additionally, it requires that the user participates in the process for disambiguation, i.e., the user must select the proper term that matches her interests among a set of proposed terms.

A novel approach for generating synthetic queries is presented in [Viej 13]. In this work, the authors argue that past queries do not always effectively reflect users' interests, and they give several reasons for this. For example, one of them is that users may circumstantially submit queries related to a certain topic which is quite far from their real interests, adding an erroneous bias to the generated profile. Another reason is that there is no pattern that defines the structure of a query and, hence, it can be quite challenging to extract accurate interests from these kinds of data.

Therefore, this work proposes: (1) to build user profiles using information from their social network accounts (e.g., Twitter) instead of from their past searches; and (2) to employ the profiles in order to generate synthetic queries related to users' interests. For this purpose, the scheme profiles users using their published tweets (i.e., text-based posts up to 140 characters published in the social network Twitter [Twit 13]). The profiles are built using the method proposed in [Viej 12b], which again applies NLP techniques (i.e., Sentence Detection, Tokenization, Part-of-speech tagging, etc.) to extract the important terms from the tweets. In a second step, the system evaluates which are the dominant and dominated categories in the profile according to their occurrence/co-occurrence frequency. Then, these categories are used as the contents of the synthetic queries to be submitted to the WSE.

Nevertheless, this system is still at an early development stage: a preliminar design is given, but more evaluations are needed in order to prove that social networks are a better source of information than past searches. Additionally, the authors assume that users have a social network and regularly publish comments about their personal interests, which may not always be true.

Distributed Systems As previously mentioned, distributed schemes require the collaboration of a group of users that work together in order to protect their privacy. Schemes in the literature that fall into this category are next described.

Crowds [Reit 98] is a system where users try to hide their actions within the actions of many others. The system puts users into a large group (the crowd) where they submit requests on behalf of other members. Users in the system are represented by processes called "jondos". Jondos are assigned to a "crowd" with other jondos by an administrative process called a "blender". The blender is also responsible for informing new jondos of other members of the crowd and for informing to all the jondos when a user joins/leaves the crowd. Besides, every node has a direct link with each node of the network (the topology is a complete graph). The communication through the link is encrypted using a key only known by the two jondos (point-to-point cryptography).

When a user wants to make a request, she establishes a random path through the network. For this purpose, she randomly picks another jondo and forwards the request to it. That jondo then flips a coin with a forwarding probability $p_f > 0.5$. Depending on the outcome of the coin flip, the jondo either selects another random jondo to which the request will be forwarded, or it forwards the request to the intended web server (the WSE). In this way, some node eventually submits the query to the WSE. Then, the results are forwarded towards the original node following the reversed path that the query used.

The security of the system relies on the fact that when a jondo receives a request, it does not know if the sender was the original requester or if it was forwarding the request for another jondo. Note that there is a tradeoff between the forwarding probability p_f and the length of the path and, hence, the query delay. The more jondos that the query visits, the higher is the query delay and the lower is the performance for the user.

- The central node represents a bottleneck in the overall system performance.
- The use of point-to-point encryption and a complete graph require that each node stores as many symmetric keys as nodes are in the network. In addition, each hop requires one encryption and one decryption. This introduces a certain overhead to the system.
- In the same way as in anonymity networks, Crowd only protects the transport of the data. Users are responsible for hiding their private information.
- Personalization is only possible if the members of the crowd share the same interests.
- This scheme is weak against the *predecessor attack* [Wrig 04]: To attack Crowds, a number of attackers may simply join the crowd and wait for paths to be reformed. Each attacker can log its predecessor after each path reformation. Let us consider a user U who wants to submit several queries. Due to the random distribution of the queries among all the nodes of the network, U will forward almost all her queries to the rest of the users. As a result, several reformations will happen and U will appear in all of them. Therefore, the attackers will log U much more often than any other node.

After a large number of path reformations, it will become clear that U is the user who is generating the queries.

In [Cast 09], the useless user profile (UUP) protocol is proposed. This scheme follows an approach quite similar to Crowds: each user who wants to submit a query will not send her own query but a query of another user instead. As a result of this behaviour, the WSE cannot generate a real profile of a certain individual. Regarding privacy concerns, the relevant point is that the users do not know which query belongs to each user. This is achieved by using cryptographic tools. The system requires two components: a central node and a client application. The central node listens to client requests. After receiving n requests, it creates a new group and sends the IP addresses and port numbers to each group member. Then, the group members establish network connections between them and start to communicate without the interaction of the central node. At the end, each user is randomly assigned one query generated by a group member. Then, she submits the query to the WSE and broadcasts the results. As a result, every group member receives the search results for the query that she generated. The scheme was tested in real conditions and results show that it provides an overhead of 5.2 seconds with a group of three users and a key length of 1024 bits.

The work presented in [Lind 10] builts upon the UUP protocol. In this case, the authors argue that the UUP protocol is not secure against dishonest internal users. Therefore, they propose a new protocol resilient against internal attackers. However, in order to do this, they employ double encryptions, which significantly increases the computation time.

Additionally, the common shortcomings of both proposals ([Cast 09], [Lind 10]) are the following:

- The groups of users must be created. This also introduces a significant delay.
- It requires a large number of users in order to provide an acceptable response time.
- Groups of users are formed at random, hence the profiles are distorted with queries that may be completely different to the real interests of their owners. According to that, personalized search is not provided, thus the quality of the search results is likely to be low.

The scheme presented in [Viej 10] employs the connections inside *social networks* to distort the user profiles. In this way, a certain user that wants to submit a query can send it directly to the WSE or forward it to a neighbour (a friend in the social network). A neighbour that receives a query has the same options: submit it to the WSE or forward it to one of her neighbours. Eventually, someone submits the query to the WSE and the results are sent back to the original user by following the inverse path that the query employed. In order to balance the load and maintain privacy, two functions are defined in the system. The first function Φ determines if a user should submit a query or forward it to a neighbour and, in the second case, it determines to which neighbour it should be forwarded. The objective of Φ is to equally distribute the queries so that the WSE cannot link a query to the real user who generated it. The second function α works as a reputation-based scheme, evaluating the selfishness of each user and punishing selfish users. This process is similar to the one introduced by Crowds. Nevertheless, in [Viej 10], a certain user is only connected with a limited set of users (her friends in the social network). Another interesting point of this proposal is that the groups of users are already generated and supported by an existing social network (e.g., Facebook). Besides, users who share the same group are assumed to be friends in real life. This increases the homogeneity of the group in terms of shared interests. Therefore, this scheme generates distorted profiles that still allow the users to get a proper service from the WSE. In addition to that, this scheme improves former solutions in terms of query delay. This proposal presents some shortcomings which are next summarized:

- The authors assume that none of the users who collaborate in a query submission will swap the correct answer from the WSE for a fake one. Some approaches to cover this point are given but it is not properly addressed.

- In case of submitting illegal queries, the authors propose a liability mechanism that enables the final sender to prove to a third party (a governmental authority) that she has not generated a certain query. This liability mechanism is based on using digital signatures and it requires the users to use their own resources in keeping certificates related to past forwarded queries.

- It achieves a query delay of 3918 ms. This time is fairly better than the one achieved by other proposals in the literature but it is still significant in comparison with directly submitting a query to the WSE (in 2010 this cost was close to 400 ms according to [Viej 10]).

The scheme presented in [Erol 11b] proposes a P2P protocol that exploits social networks in order to protect the privacy of the users of WSEs. The paper presents a new version of the protocol presented in [Viej 10] with some improvements. Firstly, the authors redesign function Φ in order to increase the level of privacy obtained by the users. In the new version, this function equitably distributes the queries in a path of length two. This means that it considers not only the direct neighbours but also the neighbours of her neighbours. Consequently, the source of a query is better hidden than in [Viej 10], since it is hidden among a group with a path length of two. Additionally, the scheme is tested using real data extracted from the AOL dataset. The results show that query delay is 4205.4 ms, slightly higher than in [Viej 10]. However, the level of privacy achieved is better: nearly 90% of the users expose less than 20% of their profile.

Nevertheless, distributed protocols that use static groups (e.g., social networks) like [Viej 10] and [Erol 11b] are more vulnerable to internal adversaries. In this scenario, attackers can exploit their knowledge about the topology, since groups of users are formed once and they rarely change. Consider the case where the neighbour of a user U is a dishonest party trying to keep track of her queries. As long as the link between them exists, the attacker will receive, with a certain probability, queries that belong to U. Furthermore, in some social networks, we can assume that the attacker and the victim share a relationship that gives to the attacker a certain knowledge about her victim. Therefore, in such cases, it would be easier for the attacker to guess when the victim is forwarding a query on behalf of another user, and when she is sending her own query.

Another example of a distributed system is the system proposed in [Domi 09a]. This scheme uses memory sectors which are shared by a group of users. These users employ the shared memory to store and read the queries and their answers. There is no connection between the users. Queries and answers are encrypted in order to provide confidentiality. This proposal does not require a trusted third party to create the groups or generate the cryptographic material. Instead of that, a simple wiki-like collaborative environment can be used to implement a shared memory sector. This scheme has the following drawbacks when applied to a WSE scenario:

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

Anonymous Channels

Schemes inside this category allow the user to send queries to a WSE in a manner that they cannot be linked to her real identity (i.e., the WSE ignores the true source of the query). The general assumption of these systems is that if the WSE does not know the true identity associated to a query, then it cannot construct any profile, and privacy is fully protected. As previously mentioned, this may significantly affect the quality of the service, since the WSE can provide no result personalization.

The approaches that fall into this category are based in the inicial concept of Chaum's mix network [Chau 81]. The general idea of mix networks is to hide the correspondence between the input and the output messages of a node using cryptography. The survey presented in [Dane 09] gives a detailed explanation of systems based on mix networks. Next, we describe some of the most important works presented there that are appliable to WSEs, together with some recent schemes not included in [Dane 09].

1. *Proxy approach.* Levine and Shields [Levi 02] define a proxy as a single server that accepts conections from an initiator I and forwards them to the responder R. The key concept is that it will deliver messages from I to R but will not disclose to R that I is the source. Instead, R will see the proxy as the source.

   Some systems like DuckDuckGo [Duck 13], Ixquick [Ixqu 13], Start Page [Star 13], PageWash [Page 13], Yippy [Yipp 13] work as intermediate nodes between the user (I) and the WSE (R): they receive the terms that must be searched, submit the queries to the WSE and show the answers to the user.

Due to this process, the user does not store any cookie from the WSE and the IP of the user is not known by the WSE.

These schemes are supposed to store neither the terms which have been submitted nor the cookies obtained in the process. Nevertheless, a proxy is not the best solution to protect the privacy of the users because profiling could be done at the proxy. Therefore, this solution requires the company that offers the service not to monitor or log the traffic of the users. This means that instead of trusting the WSE, users have to trust the proxy company.

2. *Web MIXes*. Web MIXes [Bert 01] is a system that provides anonymous and unobservable real-time Internet access. Regarding similar proposals, it incorporates an authentication mechanism that prevents flood attacks. Additionally, it also includes a feedback system with an interface that informs users about their current level of protection.

   The Web MIXes architecture consists of three parts concatenated in order to form an anonymous tunnel:

   (a) *The Java Anon Proxy (JAP).* It is a program installed in the user's computer that prepares the data to be later anonymized in the network. Its functions include user's registration in the system and communication between network and application layer.

   (b) *The MIXes.* They are computers connected via the Internet forming a logical chain. The first MIX receives data sent by the JAP. Then, every MIX in the chain scrambles the order of data streams and changes their coding using cryptography to make traffic correlation attacks difficult, and sends data to the next MIX.

   (c) *Cache-proxy.* It receives data from the last MIX. Then, it sends it to the Internet and receives the answers from the servers (in our scenario, the results from the WSE). The answers are sent back to the user via the MIXes (in reverse order).

   The work presented in [West 10] evaluates this protocol in front of several kinds of adversaries. The analysis reveals a flaw in the authentication phase, allowing an external attacker to perform a *replay attack* (i.e., a data transmission that is maliciously repeated or delayed). Another disadvantage of Web MIXes is outlined in [Bohm 04], where it is argued that Web MIXes work in a synchronous fashion, which is not well adapted for the asynchronous nature of widely deployed TCP/IP networks.

3. *Onion routing techniques.* Onion routing [Gold 99] is a technique to establish anonymous channels that preserve the privacy of the users. The authors in [Sain 07] propose to use the anonymous channels of onion routing to submit queries to the WSE.

   In the onion routing-based systems, there is a set of servers called onion routers that relay traffic for users. Let $R = R_1, R_2, \ldots, R_n$ represent a set of n onion routers in the network. Each onion router maintains a private key and a public key. The public key is known by the users of the system.

   At the first step, the user constructs a multiply encrypted tunnel, called circuit, through the network. For this purpose, the user selects an ordered sequence of onion routers in the network to use as the path of the circuit.

   Then, the user randomly generates two symmetric secret keys (a forward key KF_i and a backward key KB_i) for each router R_i along the path. It also defines forward and backward cryptographic functions f_i and f_i^{-1}, respectively. The pair (KF_i, f_i) is used to encrypt data in the path from the user towards the receiver. The pair (KB_i, f_i^{-1}) is applied to the data that travels from the receiver to the initial user.

   For example, if a user chooses the path R_x, R_y, R_z, she would construct the "onion" as follows:

$$\begin{aligned} & E_x(t_x, f_x, KF_x, f_x^{-1}, KB_x, R_y, \\ & \quad E_y(t_y, f_y, KF_y, f_y^{-1}, KB_y, R_z, \\ & \quad\quad E_z(t_z, f_z, KF_z, f_z^{-1}, KB_z, \emptyset))) \end{aligned}$$

   Where t_i is the expiration time of the message, and $\emptyset$ indicates to R_z that it is the last router in the path.

   The user sends the onion to the first router (R_x), which removes the outermost layer of the encryption using its private key, and learns the symmetric keys KF_x and KB_x, as well as who the next router of the path (R_y) is. Each router along the path repeats this process, until the onion reaches the end of the path.

Similarly, the response is sent along the reverse path of the circuit, using the backward key KB_i and the function f_i^{-1}.

The most renowned implementation of the onion routing technique is the Tor Project [Tor 13]. Tor was presented as the second-generation Onion routing technique, adding perfect forward secrecy, congestion control, directory servers, integrity checking, configurable exit policies, and a practical design for location-hidden services via rendez-vous points. The novelty of Tor is that, instead of using the 'onion', the client connects directly to the first node, then it requests this node to connect to the next one, and so on. The main advantage of Tor is its usability, also enhanced by several Firefox plugins [Foxt 13, Torb 13] which allow to use Tor in a simple way.

Since it was presented, Tor has been widely studied in the literature and adapted for several purposes. Some of the most recents works based on Tor are [Dosh 13], [Cata 13], and [Cast 13]. For example, [Dosh 13] improves the overhead and the overall security strength of Tor, and it also provides failure tolerance in case that an onion router in the communication path breaks down. On the other hand, [Cata 13] outperforms TOR by achieving forward secrecy in a fully non-interactive way (i.e., making the circuit-building non-interactive, achieving linear round complexity and non-interaction among routers, users and the Key Generation Center during the periodic routers' key update). According to the presented experimental results, this allows to create the network of onion routers with a slimmer and faster management. Regarding [Cast 13], it proposes a new circuit selection algorithm that allows Tor to provide enough performance for low-latency services.

On the other hand, several attacks to Tor have also been found. For example, as stated in [Dane 09], Tor does not attempt to offer security against even passive global observers of a circuit. Furthermore, the work presented in [Syve 11] analyzes some of the threats against which Tor currently offers only limited protection. These threats happen because by using Tor, some features of the communication are still apparent to an observer. If an adversary can see both ends of a Tor circuit, he can trivially correlate who is talking to whom over that circuit. This is generally known as an *end-to-end correlation attack*. In the WSE scenario, these kinds of attacks seriously threaten users' privacy, since the WSE could link a user with the query that she generated.

Similar to Tor, the Invisible Internet Project (I2P) [I2P 13] builds an anonymous network layer designed to be used for anonymous communication. The application that implements this layer is called an *I2P router*, and the computer who executes it is called an *I2P node*. Following the onion routing scheme, messages are wrapped with several layers of encryption, and the network is distributed and dynamic. In [I2P 13] several advantages of I2P over Tor can be found. For example, it is stated that unlike circuits in Tor, tunnels in I2P are short lived, decreasing the number of samples that can be used to perform an attack. Moreover, I2P employs unidirectional tunnels instead of Tor bidirectional circuits, doubling the number of nodes an attacker has to compromise to get the same information.

2.2 Discussion

In this section, we introduce a further analysis of the previously described client-side privacy enhancing technologies. First of all, we define a common evaluation framework for these systems. This framework includes properties which are used to evaluate users' requirements and acceptance related to any privacy-enhancing technology for web search. Then, all the systems considered in this chapter are studied and compared according to the properties included in the framework.

2.2.1 Common Evaluation Framework

Many different approaches can be used to evaluate and compare privacy enhancing technologies for web search. For example, systems can be evaluated according to the method employed to obtain privacy, or to the characteristics of the profile that the WSE obtains, or to the type of adversaries that security takes into account, etc. However, since this dissertation focuses on the client-side perspective, in this section we consider an approach that evaluates some points related to users' experience and the quality of the service that they receive. This means that technical aspects of the protocols (e.g., key lengths, required number of users, resilience against specific attacks, etc.) are not considered in this section, since they have already been thoroughly described in Section 2.1.

In order to delimitate which properties are considered in the framework, we have included all the user's requirements presented in the set of papers reviewed in this chapter. Note that these properties are not always denoted by the same name, and that sometimes they are not explicitly defined in one of the papers in the considered set, but they are remarked as desirable or employed to criticize other works in the field.

Next, the studied properties as well as the papers where they appear are listed and defined.

1. **Performance**. It appears as an important user requirement in most papers ([Aram 12], [Bert 01], [Bohm 04], [Cast 09], [Cast 13], [Domi 09a], [Erol 11b], [Levi 02], [Lind 10], [Reit 98], [Viej 10], [Viej 12a], [Viej 13], [Sanc 13], [Sain 07]). In the WSE scenario, the performance is defined as the query delay, i.e., the time spent since the user generates the query until she receives the results. Normally, privacy-preserving systems introduce an overhead over traditional web search. However, this delay should be reasonable enough for the system to be widely adopted.

2. **Quality of retrieved results**. Another common requirement in many papers ([Aram 12], [Domi 09b], [Erol 11b], [Leun 12], [Xu 07], [Viej 10], [Viej 12a], [Viej 13], [Shen 05], and [Sanc 13]) is to balance the trade-off between privacy and quality of retrieved results. Many of the reviewed systems alter the user profiles that the WSE owns at some point. This may change the ranking of results that the WSE returns, thus making the personalization less accurate.

3. **Plausible deniability**. This property, applied to WSE, is defined as the lack of sufficient evidence to prove that a query was generated by a particular user. This property is also referred to as *liability*, and it appears in some of the studied papers: [Aram 12], [Bert 01], [Levi 02], [Muru 09], [Reit 98], [Viej 10], and [Viej 12a]. A particularity of this property is that some works argue that it should be enhanced, while others design mechanisms to prevent it. The works that try to prevent plausible deniability are mainly distributed systems (see Section 2.1.2), where it is desirable that honest users can prove to a third party that they have not generated a certain query (e.g., a query that breaks the law) but they were only forwarding it on behalf of another user.

4. **Availability**. This property means that any user can employ the system and privately retrieve the results at any time. The papers where this property appears are: [Bohm 04], [Erol 11b], [Hoch 02], [Reit 98], [Ye 09], [Viej 10],

and [Viej 12a].

5. **User interaction**. It refers to the extent to which the user is required to participate in the system. The user may be required to configure some parameters or to explicitly select her interests through some kind of interface. The papers that speak about this property are: [Bert 01], [Cran 06], [Erol 11b], [Hoch 02], [Reit 98], and [Sain 07].

6. **Tolerance to complex queries**. Some papers like [Erol 11b], [Viej 10], and [Sanc 13], manifest the need for accepting any kind of query that a user may generate. Complex queries are formed by several terms or sentences. Therefore, some works, especially those that generate synthetic queries from real queries, may not be compliant with this property.

7. **Incentives to cooperate**. In systems where users interact with other entities, it may be necessary to incentivize them to cooperate, as it is remarked in papers like [Bohm 04], [Erol 11b], [Reit 98], and [Viej 10]. Without these incentives or rewards, selfish users may cause the system to fail in practice.

8. **User storage**. Some systems may require users to locally store a great amount of information. According to papers like [Domi 09a] and [Sanc 13], it is another property that must be considered when evaluating users' experience.

2.2.2 Analysis of the properties

Table 2.1 shows the properties defined above for each of the papers that propose a system to protect privacy, studied in Section 2.1. Note that, as an exception, we have decided not to include PIR because, as previously remarked, it cannot be applied to the WSE scenario and, hence, there is no reason to study it in this section. Regarding the rest of proposals, the table shows that most properties are homogeneous inside a group, i.e., systems that belong to the same category or sub-category behave similarly regarding some properties.

For example, *performance* and *availability* are properties that decrease when users depend on an external entity in order to submit the query and receive the results. Distributed systems and anonymous channels are clear examples of this situation. In both cases, forwarding the query to other entities significantly increases the response time (i.e., performance decreases) and if the external entities

Table 2.1: Client-side PETs: summary of properties.

				Performance	**Quality of Retrieved Results**	**Plausible Deniability**	**Availability**	**User interaction**	**Complex queries**	**Incentives to cooperate**	**User storage**
Collaborative			P3P [Cran 06]	↑	↑	✗	↑	↑	✓	n/a	↓
			Do-Not-Track [C So 12]	↑	↑	✗	↑	↑	✓	n/a	↓
			UCAIR [Shen 05]	↑	↓	✗	↑	↑	✓	n/a	↑
			[Xu 07]	↑	↓	✗	↑	↑	✓	n/a	↑
			[Leun 12]	↑	↓	✗	↑	↑	✓	n/a	↑
			[Kram 13]	↑	↓	✗	↑	↑	✓	n/a	↑
Non-collaborative	Obfuscation techniques	Standalone	TrackMeNot [Trac 13]	↑	↓	✓	↑	↓	✓	n/a	↓
			GooPIR [Domi 09b]	↑	↓	✓	↑	↓	✗	n/a	↑
			PDS [Muru 09]	↑	↓	✓	↑	↓	✓	n/a	↑
			[Viej 12a]	↑	↑	✓	↑	↓	✗	n/a	↑
			[Sanc 13]	↑	↑	✓	↑	↓	✓	n/a	↑
			QueryScrambler [Aram 12]	↑	↓	✓	↑	↓	✗	n/a	↑
			[Viej 13]	↑	↑	✓	↑	↓	✓	n/a	↑
		Distributed	Crowds [Reit 98]	↓	↓	✓	↓	↓	✓	↓	↑
			UUP [Cast 09]	↓	↓	✓	↓	↓	✓	↓	↓
			[Lind 10]	↓	↓	✓	↓	↓	✓	↓	↓
			[Viej 10]	↓	↑	✗	↓	↓	✓	↑	↑
			[Erol 11a]	↓	↑	✗	↓	↓	✓	↑	↑
			[Domi 09a]	↓	↓	✓	↓	↓	✓	↓	↓
	Anonymous channels		Proxies [Levi 02]	↓	↓	✓	↓	↓	✓	↓	↓
			WebMIXes [Bert 01]	↓	↓	✓	↓	↓	✓	↓	↓
			TOR [Tor 13]	↓	↓	✓	↓	↓	✓	↓	↓
			[Dosh 13]	↓	↓	✓	↓	↓	✓	↓	↓
			[Cata 13]	↓	↓	✓	↓	↓	✓	↓	↓
			[Cast 13]	↓	↓	✓	↓	↓	✓	↓	↓
			I2P [I2P 13]	↓	↓	✓	↓	↓	✓	↓	↓

are busy or suffer from technical problems (for example, a proxy that has crashed), users cannot privately submit their queries (i.e., availability decreases).

Table 2.1 also shows that no collaborative system provides *plausible deniability*. In these systems, the query is directly submitted to the WSE, there is no external entity and no synthetic query to obfuscate it. Therefore, a user cannot *deny* that she generated a particular query. On the other hand, non-collaborative systems (except [Viej 10] and [Erol 11a]) provide *plausible deniability*. Obfuscation techniques add noise to the real query: a user can *deny* that she generated a particular query by arguing that it was a synthetic query automatically generated (standalone schemes) or that she submitted it on behalf of another user (distributed schemes). There are two exceptions, since [Viej 10] and [Erol 11a] employ a *liability mechanism* to prevent a dishonest user to submit a query that breaks the law. In these systems, the anonymity of a user is revocable in front of a governmental authority and, in this case, the dishonest user lacks from *plausible deniability*. Finally, anonymous channels are also considered to provide *plausible deniability* in front of the WSE because a user is never linked to its query and the source cannot be found.

Collaborative systems require higher *user interaction*. Some of them (P3P [Cran 06], Do-Not-Track [C So 12]) require the user to know and define the policies that should be applied in their searches. For the others (UCAIR [Shen 05], [Xu 07], [Leun 12], [Kram 13]), the user is required to adjust some parameters (e.g., the level of detail that will be revealed to the WSE). Some non-collaborative systems like TrackMeNot [Trac 13], WebMIXes [Bert 01], and TOR [Tor 13] also include a user interface. However, this interface is mainly used to provide information about the system, but does not demand the user to actively participate in the process.

Regarding *complex queries*, all the systems but GooPIR [Domi 09b], [Viej 12a] and QueryScrambler [Aram 12] allow the user to submit any kind of query. GooPIR [Domi 09b] uses a Thesaurus and hence, only single words are supported. As explained in Section 2.1.2, [Viej 12a] do not work well with queries that have multiple words. QueryScrambler [Aram 12] employs hypernymy and hyponymy relations in WordNet, which cannot be applied either to multiple words, or to adverbs or adjectives.

The *incentives to cooperate* are only appliable to systems where users forward their query to a third party, i.e., distributed systems and anonymous channels. Among distrubted systems, the only ones that provide mechanisms to incentivize their users are [Viej 10], and [Erol 11a]. In [Viej 10] and [Erol 11a], if a user does not accept queries from her neighbours, there is an heuristic that causes her neighbours to refuse her queries. Regarding anonymous channels, the users in the network can *volunteer* as a relay. However, this is not a requirement, and no incentive to cooperate is given.

The properties *quality of the retrieved results* and *user storage* are more heterogeneous inside the groups and they are therefore described for each system separately. For example, the quality of the results for P3P [Cran 06] and Do-Not-Track [C So 12] is the same as if the user were not using any tool, since the user profile that the WSE owns is never modified. Additionally, they do not require a high user storage, since P3P [Cran 06] only needs to store a file with the selected policies, and Do-Not-Track [C So 12] is simply a HTTP header.

On the other hand, with UCAIR [Shen 05], [Xu 07], [Leun 12], and [Kram 13], the quality of the results may often be lower. As explained in Section 2.1.1, the re-ranking is done at the client side, and this is less effective than in the WSE side: since the client has no global index for all websites, a general retrieval function cannot be applied. Moreover, these systems require to store a higher amount of information, including the user profile, the search context and history information (e.g., the submitted queries and clicked results).

Standalone systems provide different quality of the results. With TrackMeNot [Trac 13], GooPIR [Domi 09b], and PDS [Muru 09], users obtain distorted profiles that do not reflect their real interests. Therefore, the WSE cannot correctly personalize the results, thus reducing their quality. On the contrary, [Viej 12a], [Sanc 13], and [Viej 13] maintain the interests of the user in the distorted profile, thus allowing a better personalization. QueryScrambler [Aram 12] is a particular case: although it maintains the interests of the user in the profile, the real query is never submitted. Consequently, the results must be re-ranked in the client side and, as explained before, this is not as accurate as if the WSE had done it. Regarding user storage, standalone systems require some knowledge source in order to generate the synthetic queries. If the source can be consulted online (like TrackMeNot [Trac 13] that employs RSS feeds), the user is not required to store a great amount of information. If the source must be downloaded to the client's machine (like the ODP that [Viej 12a], [Sanc 13], and [Viej 13] recommend to download, or the Thesaurus employed by GooPIR [Domi 09b], or other Latent Semantic Indexing used by PDS [Muru 09], or the WordNet that QueryScrambler employs), then the user storage increases.

In distributed systems, users submit queries generated by other group members. If these group members share the same interests (as it is assumed by [Viej 10] and [Erol 11a]), the profile will maintain the interests of the user, also maintaining the quality of the retrieved results. Contrarily, if the users are randomly grouped (as in Crowds [Reit 98], UUP [Cast 09], [Lind 10], and [Domi 09a]), the real interests are mixed with other random interests, and the WSE cannot perform an accurate personalization. Concerning the user storage in distributed systems, [Lind 10], and [Domi 09a] do not require the user to store great amounts of data. However, Crowds [Reit 98] requires a user to store as many symmetric keys as nodes are in the network, which represents a significant amount of data. The systems proposed in [Viej 10] and [Erol 11a] store certificates as a proof of the query transaction between two users (this is a part of the liability mechanism). This means that, for each forwarded query, a user must store some information signed with a secret key, and keep it in a safe place.

Finally, when a user employs any anonymous channel, she obtains no profile, i.e., the WSE knows neither the user nor her interests. Therefore, no personalization can be performed, and the quality of the results is reduced. As for user storage, none of these systems specify that great amounts of data should be stored in the client side.

Regarding the overall set of properties, Table 2.1 shows that there is no scheme that perfectly complies with all the properties. However, the systems that better improve the users' experience are [Sanc 13] and [Viej 13], since they only lack of one compliance: they need to store the ODP ontology, which represents a great amount of data. Despite requiring a high user storage, both proposals provide high performance, high quality of the retrieved results, plausible deniability, high availability, low user interaction, and allowance of complex queries.

2.3 Conclusions of the state of the art

In this chapter, we have analyzed the state of the art in client-side privacy enhancing technologies for web search. First, the study presents a way to classify the literature of this field. Then, it describes how existing proposals work and also their advantages and disadvantages. Finally, a common evaluation framework to evaluate and compare these proposals from the user's experience point of view is presented. In particular, this framework includes eight properties considered relevant in this research field.

The matter of privacy-preserving in web search is too vast to cover in a single work. As this study shows, there are many privacy-preserving alternatives, and the techniques applied in collaborative and non-collaboratives approaches (and their subclassifications) are very different. We believe that collaborative approaches excessively rely on the WSE, and hence, we decided to focus on non-collaborative approaches. Moreover, inside non-collaborative approaches, this dissertation only focuses on distributed obfuscation techniques. The reason for this choice is that both standalone obfuscation techniques and anonymous channels are already at an advanced state of development, while distributed obfuscation approaches still leave room for improvement. More specifically, regarding distributed schemes, our work focuses on the following points of improvement:

1. *Response time:* One of the constraints when employing a group of users that exchange their queries is the time required until they receive the results. If this delay is too high, users will be more reluctant to use the system.

2. *Level of security:* In client-side private web search, the main objective is to preserve users' privacy in front of the WSE. However, in distributed systems, malicious users may also participate in the protocol. Therefore, another point of improvement is to protect users' privacy in front of other members of the group.

3. *Quality of the retrieved results:* Section 2.1.2 describes how the semantics of the queries employed to obfuscate a profile affect the level of personalization that the WSE can provide the user. Therefore, one of the points of improvement of distributed techniques is to group users that share similar interests, so that the exchange of queries inside the group still allows a maximum of WSE personalization.

CHAPTER 3

Cryptographic Background

This chapter introduces the cryptographic background, assumptions and definitions necessary to understand our contributions described in subsequent chapters.

Contents

First of all, the n-out-of-n threshold ElGamal encryption is explained (Section 3.1). Then, a permutation network named OAS-Benes is introduced in Section 3.3. Finally, two zero-knowledge proofs called Plaintext Equivalence Proof (PEP) and Disjunctive Plaintext Equivalence Proof (DISPEP) are described in Section 3.4 and Section 3.5, respectively.

3.1 n-out-of-n threshold ElGamal encryption

In cryptographic multi-party protocols, some operations must be computed jointly by different users. In an n-out-of-n threshold ElGamal encryption [Desm 90], n users share a public key y and the corresponding unknown secret key α is divided into n shares α_i. Using this protocol, a certain message m can be encrypted using the public key y and the decryption can be performed only if all n users collaborate in the decryption process. Key generation, encryption and decryption process are next described.

3.1.1 Key generation.

First, a large random prime p is generated, where $p = 2q + 1$ and q is a prime number too. Also, a generator g of the multiplicative group $\mathbb{Z}_q^*$ is chosen.

Then, each user generates a random private key $\alpha_i \in \mathbb{Z}_q^*$ and publishes $y_i = g^{\alpha_i}$. The common public key is computed as $y = \prod_{i=1}^{n} y_i = g^{\alpha}$, where $\alpha = \alpha_1 + \ldots + \alpha_n$.

3.1.2 Message encryption.

Message encryption can be performed using the standard ElGamal encryption function [ElGa 85]. Given a message m and a public key y, a random value r is generated and the ciphertext is computed as follows:

$$E_y(m, r) = c = (c1, c2) = (g^r, m \cdot y^r)$$

3.1.3 Message decryption.

Given a message encrypted with a public key y, $E_y(m, r) = (c1, c2)$, user U_i can decrypt that value as follows:

Each user $j \neq i$ publishes $c1^{\alpha_j}$. Then, U_i can recover message m in the following way:

$$m = \frac{c2}{c1^{\alpha_i}(\prod_{j \neq i} c1^{\alpha_j})}$$

This decryption can be verified by each participant by performing a proof of equality of discrete logarithms [Chau 92].

3.2 ElGamal re-masking

The re-masking operation performs some computations over an encrypted value. In this way, its cleartext does not change but the re-masked message is not linkable to the same message before re-masking.

Given an ElGamal ciphertext $E_y(m, r)$, it can be re-masked by computing [Abe 99]:

$$E_y(m, r) \cdot E_y(1, r')$$

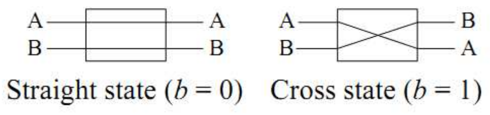
For $r' \in \mathbb{Z}_q^*$ randomly chosen and where $\cdot$ stands for the component-wise scalar product (ElGamal ciphertext can be viewed as a vector with two components). The resulting ciphertext corresponds to the same cleartext m.

3.3 Optimized Arbitrary Size (OAS) Benes

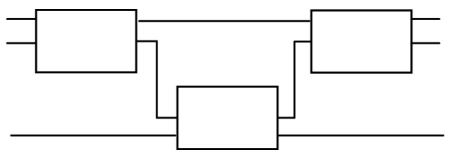
A Benes permutation network (PN) is a directed graph with N inputs and N outputs, denoted as $PN^{(N)}$. It is able to realize every possible permutation of N elements.

A Benes PN is composed by a set of 2 x 2 switches. These switches have a binary control signal $b \in \{0, 1\}$ which determines the internal state and, hence, the output. The two possible states of a 2 x 2 switch are depicted in Figure 3.1(a).

The problem with a Benes PN is that the size of the network must be a power of 2. In order to have an Arbitrary Sized (AS) Benes network, it is necessary to introduce a 3 x 3 network like Figure 3.1(b) shows. Using 2 x 2 switches and 3 x 3 networks recursively it is possible to construct a network of any size.



(a) States of a 2 x 2 switch



(b) 3 x 3 network

Figure 3.1: Basic elements of an OAS-Benes

Optimized Arbitrary Size (OAS) Benes is an extension of AS Benes that reduces the number of necessary switches in the network. The way of constructing the OAS-Benes depends on the parameter N:

- If N is even, the OAS-Benes $PN^{(N)}$ is built recursively from two even OAS-Benes of $\frac{N}{2}$-dimension called sub-networks. The sub-networks are not directly connected to the inputs and outputs. Instead of that, they are connected to $N - 1$ input-output switches, as Figure 3.3 shows.
- If N is odd, the OAS-Benes $PN^{(N)}$ is composed by an upper $\lfloor \frac{N}{2} \rfloor$ even OAS-Benes, and a lower $\lceil \frac{N}{2} \rceil$ odd OAS-Benes. The sub-networks are not directly connected to the inputs and outputs. In this case, the first $N - 1$ inputs are connected to $\lfloor \frac{N}{2} \rfloor$ switches, and the first $N - 1$ outputs are connected to $\lfloor \frac{N}{2} \rfloor$ switches. Figure 3.3 illustrates this construction.

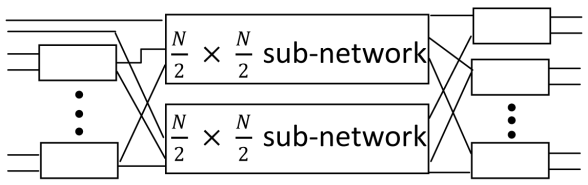

$\frac{N}{2} \times \frac{N}{2}$ sub-network
$\frac{N}{2} \times \frac{N}{2}$ sub-network

Figure 3.2: Construction of OAS-Benes for even N

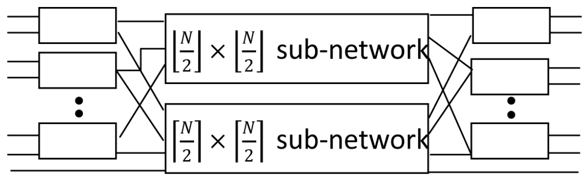

$\lfloor\frac{N}{2}\rfloor \times \lfloor\frac{N}{2}\rfloor$ sub-network
$\lceil\frac{N}{2}\rceil \times \lceil\frac{N}{2}\rceil$ sub-network

Figure 3.3: Construction of OAS-Benes for odd N

According to the way that an OAS-Benes is constructed, it is possible to account the minimum number of switches required to satisfy a permutation of N elements. The formula to calculate the minimum number of switches is:

$$S(N) = \begin{cases} (N-1) + 2 * S(\frac{N}{2}) & if\ N\ is\ even \\ \\ 2 * \lfloor\frac{N}{2}\rfloor + S(\lceil\frac{N}{2}\rceil) + S(\lfloor\frac{N}{2}\rfloor) & if\ N\ is\ odd \end{cases}$$

$$Where\ S(1) = 0,\ S(2) = 1,\ S(3) = 3$$

3.3.1 Multi-party OAS-Benes.

OAS-Benes can be used to perform a joint permutation. This means that the switches of the OAS-Benes can be distributed among a group of n users trying to realize a permutation of N inputs. However, this must be done in such a way that no user knows the overall permutation between the inputs and the outputs.

According to [Soo 02], a secure permutation (where no user knows the overall permutation) requires minimally t OAS-Benes $PN^{(N)}$, where t depends on the minimum number of honest users that the system requires. The t OAS-Benes $PN^{(N)}$ are fairly divided in n adjacent stages. Then, stage i (for $i \in 1, \ldots, n$) is assigned to user i.

In order to obtain a secure permutation, the condition that must be satisfied is that the honest users control, at least, $S(N)$ switches. For example, consider

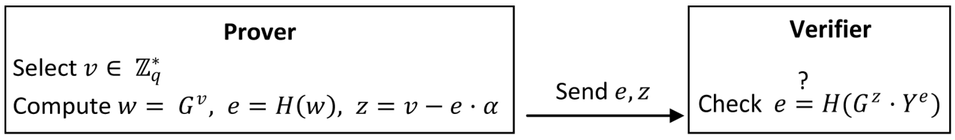

Prover
Select $v \in \mathbb{Z}_q^*$
Compute $w = G^v,\ e = H(w),\ z = v - e \cdot \alpha$
Send e, z
Verifier
Check $e \overset{?}{=} H(G^z \cdot Y^e)$

Figure 3.4: PEP protocol

a scenario with $n = 6$ users, $N = 8$ inputs and, at least, $\lambda = 3$ honest users. The number of switches of one OAS-Benes $PN^{(8)}$ is $S(8) = 17$. According to [Soo 02], the $\lambda = 3$ honest users must control 17 or more switches. This means that every user must control $\lceil \frac{17}{3} \rceil = 6$ switches. Therefore, the scheme needs at least $(6\ \textit{switches per user} \times 6\ \textit{users}) = 36$ switches that will be fairly divided among the n users. Consequently, the system requires $t = \lceil \frac{36}{17} \rceil = 3$ OAS-Benes $PN^{(8)}$.

We propose formula 3.1 in order to calculate the number of OAS-Benes required in a scheme with n users, N inputs, and λ honest users.

$$t = \left\lceil \frac{n \cdot \left\lceil \frac{S(N)}{\lambda} \right\rceil}{S(N)} \right\rceil \tag{3.1}$$

3.4 Plaintext Equivalence Proof (PEP)

PEP [M Ja 99] is an honest-verifier zero-knowledge proof protocol based on a variant of the Schnorr signature algorithm [Schn 91]. The purpose of this protocol is to prove that two different ciphertexts are the encryption of the same message.

Two ElGamal ciphertexts $(c1_a, c2_a) = (g^{r_a}, m_a \cdot y^{r_a})$ and $(c1_b, c2_b) = (g^{r_b}, m_b \cdot y^{r_b})$ for some r_a, $r_b \in \mathbb{Z}_q^*$ are plaintext equivalent if $m_a = m_b$. Let:

- $\alpha = r_a - r_b$
- $k = H(y, g, c1_a, c2_a, c1_b, c2_b)$, where $H(\cdot)$ is a hash function.
- $G = g \cdot y^k$
- $Y = \frac{c1_a}{c1_b} \cdot (\frac{c2_a}{c2_b})^k = (g \cdot y^k)^\alpha$

In order to prove that $(c1_a, c2_a) \equiv (c1_b, c2_b)$, the prover must demonstrate knowledge of α by executing the protocol of Figure 3.4.

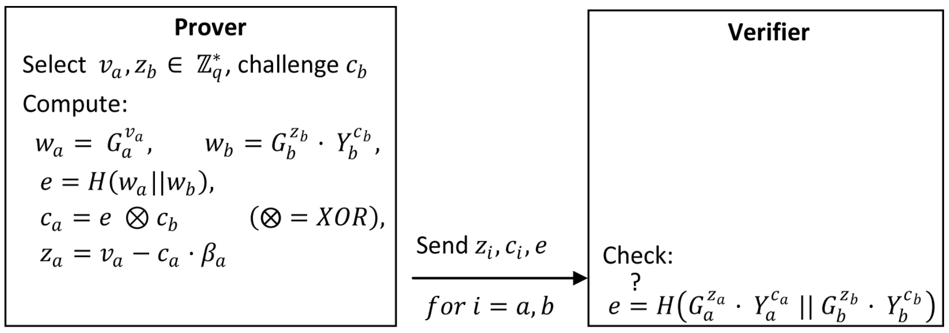

Prover
Select $v_a, z_b \in \mathbb{Z}_q^*$, challenge c_b
Compute:
$w_a = G_a^{v_a}$, $w_b = G_b^{z_b} \cdot Y_b^{c_b}$,
$e = H(w_a || w_b)$,
$c_a = e \otimes c_b$ $(\otimes = XOR)$,
$z_a = v_a - c_a \cdot \beta_a$
Send z_i, c_i, e
$for\ i = a, b$
Verifier
Check:
$e \overset{?}{=} H(G_a^{z_a} \cdot Y_a^{c_a} || G_b^{z_b} \cdot Y_b^{c_b})$

Figure 3.5: DISPEP protocol

3.5 Disjunctive PEP (DISPEP)

DISPEP [M Ja 99] is an extension of the PEP protocol. In this case, a prover demonstrates that one of two different ciphertexts is a re-masked version of another ciphertext.

Let $(c1_a, c2_a) = (g^{r_a}, m_a \cdot y^{r_a})$ and $(c1_b, c2_b) = (g^{r_b}, m_b \cdot y^{r_b})$ be two different ElGamal ciphertexts. Then, one of them is a re-masking of another ciphertext $(c1, c2) = (g^r, m \cdot y^r)$ for some r_a, r_b, $r \in \mathbb{Z}_q^*$ if $m_a = m$ or $m_b = m$. For $i \in \{a, b\}$, let:

- $\beta_i = r - r_i$
- $k_i = H(y, g, c1, c2, c1_i, c2_i)$
- $G_i = g \cdot y^{k_i}$
- $Y_i = \frac{c1}{c1_i} \cdot (\frac{c2}{c2_i})^{k_i} = (g \cdot y^{k_i})^{\beta_i}$

In order to prove whether $m_a = m$ or $m_b = m$, the prover must demonstrate knowledge of β_i by executing the protocol of Figure 3.5. Without loss of generality, in Figure 3.5 we assume that the prover is showing $m_a = m$.

CHAPTER 4

Improving Query Delay in Private Web Search

This chapter introduces a distributed protocol which is an improvement of the Useless User Profile (UUP) protocol, reducing the query delay and incentivizing users to follow the protocol in order to protect their privacy.

Contents

We introduce the novelty of the approach in §4.1. Section 4.1.1 introduces the protocol proposed in [Cast 09], detailing the parts of the system that the new protocol employs later. Section 4.2 explains the new protocol. In Section 4.4 a privacy analysis is performed. The implementation and simulation of the protocol is reported in Section 4.5. We also compare the results obtained in the new protocol with the results presented for the UUP protocol. Finally, Section 4.6 gives some conclusions

4.1 Novelty of the approach

This chapter focuses on the first point of improvement (*response time*) described in Section 2.3. Current multi-party schemes for private web search in the literature significantly increase the query delay. This is the time that users have to wait in order to obtain the search results for their queries. In this chapter, we present a modification of the Useless User Profile (UUP) protocol. The resulting scheme has been tested in an open environment and the results show that it achieves the lowest query delay which has been reported in the literature. In addition to that, it incentivizes users to follow the protocol in order to protect their privacy.

4.1.1 UUP protocol overview

The UUP protocol includes a central node that receives requests from users. When it has n requests, it creates a new group. The users of this group build a group key (see Section 3.1). Then, every user encrypts her query with the group key and broadcasts it.

The set of encrypted queries is sequentially forwarded from one user to the next until the last one.

In her turn, each user remasks and permutates the order of the encrypted queries, and sends the result to the next user. The last user obtains and broadcasts the set of encrypted queries that cannot be linked to the original set.

At the end, the users reveal their shares of the group key in order to decrypt the queries. Every user submits one decrypted query to the WSE and broadcasts the response to all the members of the group.

Note that the UUP protocol assumes that all the users follow the protocol. It also assumes that there are no collusion between the entities that participate in the protocol.

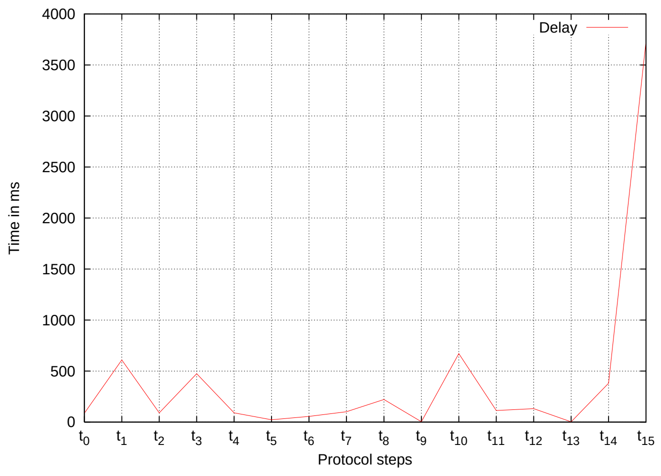


Figure 4.1: Partial times of the *UUP protocol* in an open environment with $n = 3$ users and $l = 1024$ bits.

4.1.2 Computation and communication cost of the UUP protocol

In order to test this protocol, the authors of [Cast 09] performed several simulations. Different parameters were tested (*i.e.* number of users, key length, type of environment –local or open–). The authors indicated that a key of 1024 bits is computationally safe. They also argued that 3 users is the best value for the size of the group. Accordingly, this chapter focuses on the tests they made in an open environment with $n = 3$ and $l = 1024$ bits.

However, two years have passed since [Cast 09] was tested. During this time, the Internet bandwidth and related resources have increased. Using the time results that appear on [Cast 09] could be detrimental to the UUP protocol in later comparisons. Therefore, it was necessary to simulate again the UUP protocol in the present conditions. This was done by executing the same source code that the authors of [Cast 09] used for their simulations. Figure 4.1 shows the obtained time results. Next, the meaning of each time interval is explained:

- t_0: time interval required to initialize the applet.
- t_1: time interval required by U_i to connect with the central node and get a response. This response includes the information needed to contact with the other members of the group and the parameters to create the group key.

- t_2: time interval required by U_i to create her group key share α_i.
- t_3: time interval required by U_i to establish a connection with the other group members.
- t_4: time interval required by U_i to broadcast her key share $y_i = g^{\alpha_i}$.
- t_5: time interval required by U_i to generate the group public key y using the shares received from all group members.
- t_6: time interval required by U_i to encrypt the query m_i using the group public key y.
- t_7: time interval required by U_i to send the resulting ciphertext c_i^0.
- t_8: time interval required by U_i to re-mask the received ciphertexts and permute them.
- t_9: time interval required by U_i to send the results which have been obtained in the re-masking/permuting step.
- t_{10}: time interval needed since user U_i sends her ciphertext c_i^0 until the last user broadcasts the ciphertexts $\{c_1, \cdots, c_n\}$. This period includes neither the time required to perform the re-masking/permuting step (t_8) nor the time required to send the ciphertexts to the next user (t_9).
- t_{11}: time interval required by U_i to calculate the shares which are used to decrypt the ciphertexts.
- t_{12}: time interval required by U_i to broadcast the shares.
- t_{13}: time interval required by U_i to decrypt the ciphertext c_i that she has received. Note that U_i needs all the shares sent by the other group users to perform this step.
- t_{14}: time interval required by the WSE to return the answer to the query m^i which has been sent by U_i.
- t_{15}: time interval required by U_i to distribute the received answer to the other users.

The increase of the Internet bandwidth and related resources benefits most of the time intervals. However, the time interval t_{15} has augmented significantly. This happens because, when the authors of [Cast 09] tested their protocol, a page of Google results took up about 50 Kilobytes. Nowadays, the size of a results page has increased up to 150 kilobytes due to the inclusion of multimedia content.

As a result, the current query delay of the UUP protocol is higher. Although the UUP protocol obtained a 5.2 second query delay when it was tested, recent simulations indicate that it obtains a 6.8 second query delay at present.

4.2 Our proposal in detail

4.2.1 Weak points of the UUP protocol

Figure 4.1 shows that $\{t_1, t_3, t_{10}, t_{15}\}$ are the time intervals which present the highest delay. Note that the most significant overhead is introduced by t_{15} (*i.e.* the broadcast of the results obtained from the WSE).

Interval t_1 refers to the connection to the central node. Interval t_3 is the connection establishment between users. These two steps are inherent to a TCP connection, hence they cannot be reduced. Instead of that, the novelty of our approach is based on two modifications:

- Restructure the steps of the protocol that have an effect upon t_{10}.
- Remove the final broadcast phase t_{15}.

In order to reduce t_{10}, the new proposal modifies the way that the queries are decrypted. In the UUP protocol, when a user U receives a list of ciphertexts, she re-masks and permutes each ciphertext, and then forwards the result to the next user. When the last user has re-masked and permuted the ciphertexts, she broadcasts the results. At this point, the queries are still encrypted. Therefore, the users have to exchange their shares, and then decrypt the ciphertexts.

Instead of this, the proposed modification consists in decrypting the queries before the broadcast. This means that the last user broadcasts the queries in cleartext, instead of ciphertexts. This has two advantages: (i) messages are broadcast faster because they are shorter; and (ii) users do not need to exchange their shares anymore.

In order to decrypt the queries before the broadcast, a new operation called *partial decryption* is used. This operation is described in Section 4.2.2. With the proposed modification, when U receives a list of ciphertexts, first of all she *partially decrypts* each ciphertext, and then she re-masks and permutes it. Only when all the users have *partially decrypted* the list of ciphertexts, the cleartexts are obtained. This means that when the last user receives the list of ciphertexts, they are still encrypted. But when she *partially decrypts* them, she obtains the queries in clear.

The second modification consists in removing the final step of the protocol (*i.e.* t_{15}).

In the new protocol, before the broadcast phase, all users know the list of n queries which have been generated by the group. Therefore, each user submits all the queries (including her own query) instead of submitting only one. Regarding performance, the main advantage is that users directly receive the answer for their question from the WSE. Hence, they do not have to send more messages. Regarding privacy, it is still protected since the WSE cannot distinguish the query that belongs to a certain user from the group of queries which have been submitted.

In addition, with this modification users protect their privacy only if they follow the protocol. In [Cast 09], a selfish user who neither decrypts the ciphertexts, nor submits a query nor broadcasts the results can still get correct results if the rest of members of the group follow the protocol properly. As a result, users have no real incentive to behave honestly. On the other hand, in the new protocol, users preserve their privacy by hiding their own queries among queries of other users. In order to get the queries of the others, users must follow all the steps of the protocol.

4.2.2 Cryptographic building blocks

Some cryptographic building blocks in our protocol are the same as the blocks used in [Cast 09]. More specifically, the re-masking operation is performed using the *ElGamal re-masking technique* described in Section 3.2. Regarding the encryption, the new protocol employs the *n-out-of-n threshold ElGamal encryption* with the same *key generation* and the same *message encryption* (see Section 3.1). As stated before, the message decryption is performed using a new method named *message partial decryption* which is now explained:

- *Message partial decryption*. The ciphertext before any partial decryption is denoted as $E_y(m,r) = (c1, c2)$. This ciphertext is encrypted with a public key $y = g^{\alpha_1+\cdots+\alpha_n}$, where α_i is the private key generated by user U_i. In order to partially decrypt the ciphertext, user U_i employs her private key as follows:

$$c2' = \frac{c2}{c1^{\alpha_i}}$$

The result of this operation is used to build another ciphertext denoted as $E_{y'}(m,r) = (c1, c2')$. In this case, the ciphertext is encrypted with a public key $y' = g^{\alpha_{(i+1)}+\cdots+\alpha_n}$.

With this operation, users can individually contribute to the decryption of the queries during the execution of the protocol. When a user partially decrypts a ciphertext, this ciphertext is no longer encrypted with her share. Thus, with the contribution of all the users, the ciphertexts are finally decrypted.

4.3 Protocol description

4.3.1 Group set up

The user U_i who wants to submit a query to the WSE, contacts the central node requesting to be included in a group. The central node is listening to user requests. Once it has n requests, a group $\{U_1, \ldots, U_n\}$ is created. Then, the central node notifies the n users that they belong to the same group. The users receive a message with the IP addresses and the ports of the other members of the group in order to establish a communication channel with them. After this step, users can send messages directly to each other and the central node is no longer needed.

4.3.2 Group key generation

1. Users $\{U_1, \ldots, U_n\}$ agree on a large prime p where $p = 2q + 1$ and q is a prime too. Next, they pick an element $g \in \mathbb{Z}_q^*$ of order q.

2. In order to generate the group key, each user U_i performs the following steps:

   (a) Generates a random number $a_i \in \mathbb{Z}_q^*$.

(b) Calculates her own share $y_i = g^{a_i} \bmod p$.

(c) Broadcasts her share y_i and receives the other shares y_j for $j = (1, \ldots, n), j \neq i$.

(d) Uses the received shares to calculate the group key:
$y = \prod_{1 \leq j \leq n} y_j = g^{a_1} \cdot g^{a_2} \cdot \ldots \cdot g^{a_n}$

4.3.3 Anonymous query retrieval

1. User U_i encrypts her query m_i:

   (a) U_i generates a random number r_i.

   (b) U_i encrypts her query m_i with the group key y:
   $c_i^0 = E_y(m_i, r_i) = (g^{r_i}, m_i \cdot y^{r_i}) = (c1_i, c2_i)$

2. For $i = (2, \ldots, n)$, each user U_i sends c_i^0 to the first member of the group (U_1).

3. For $i = (1, \ldots, n-1)$, each user U_i performs the following operations:

   (a) Receives the list of ciphertexts $\left\{c_1^{i-1}, \ldots, c_n^{i-1}\right\}$.

   (b) Using her share of the group key, partially decrypts the list of ciphertexts using the algorithm described in Section 4.2.2. The resulting list of ciphertexts is denoted as $\left\{c_1^{i-1'}, \ldots, c_n^{i-1'}\right\}$.

   (c) The list of ciphertexts $\left\{c_1^{i-1'}, \ldots, c_n^{i-1'}\right\}$ is re-masked using the re-masking algorithm described in Section 3.2 with a key $y' = \prod_{j=i+1}^{n} g^{\alpha_j}$. As a result, U_i obtains a re-encrypted version $\left\{e_1^{i-1}, \ldots, e_n^{i-1}\right\}$.

   (d) Permutes the order of the ciphertexts at random, obtaining a reordered version $\left\{e_{\sigma(1)}^{i-1}, \ldots, e_{\sigma(n)}^{i-1}\right\}$

   (e) Sends the list of ciphertexts $\left\{c_1^{i}, \ldots, c_n^{i}\right\} = \left\{e_{\sigma(1)}^{i-1}, \ldots, e_{\sigma(n)}^{i-1}\right\}$ to U_{i+1}.

4. The last user U_n performs the following operations:

   (a) Receives the list of ciphertexts $\left\{c_1^{i-1}, \ldots, c_n^{i-1}\right\}$.

   (b) Using her share of the group key, partially decrypts the list of ciphertexts using the algorithm described in Section 4.2.2. At this point, U_n owns the cleartexts of the queries.

   (c) Broadcasts the queries to the rest of users $\{U_1, \ldots, U_{n-1}\}$.

4.3.4 Query submission and retrieval

1. Each group member U_i submits the n received queries to the WSE.
2. Each user only takes the answer that corresponds to her original query.

4.4 Privacy analysis

In this section, the privacy of the system in the presence of dishonest entities is analyzed. These entities are: a dishonest user, a dishonest central node and a dishonest WSE.

4.4.1 Dishonest user

Similarly to [Cast 09], in order to guarantee the correctness of the process, our protocol assumes an scenario where the users follow the protocol and there are no collusions. Nevertheless, the work presented in [Lind 10] modifies the scenario of [Cast 09] introducing malicious internal users. More specifically, [Lind 10] indicates three attacks that malicious internal users can do in order to learn the queries of other participants. We next detail how the proposed protocol behaves against the three attacks and how to modify it to work in the scenario of [Lind 10]:

1. **Stage-skipping attack**: In this attack, the last user U_n remasks and permutes the original list of ciphertexts instead of the list received from U_{n-1}. After the decryption, U_n knows which query belongs to each user.

   This attack was a threat in the UUP protocol, but it cannot be conducted in our protocol. This happens because our protocol employs the partial decryption operation. The original list of ciphertexts is encrypted under the group key while the list received from U_{n-1} is only encrypted under the key of U_n. User U_n cannot remove the encryptions under the shares of the other members of the group. Consequently, U_n cannot obtain the cleartexts from the original list of ciphertexts.

2. **Input-replacement attack**: In this attack, the first user U_1 learns the query of one of her partners. Initially, U_1 receives the original list of ciphertexts and removes all the ciphertexts except the one that belongs to her victim. Then, U_1 replaces the removed ciphertexts with individually remasked copies of her victim's ciphertext (or with encryptions of keywords chosen by U_1).

When the last user broadcasts the queries, U_1 is able to identify which query belongs to her victim.

In order to avoid this attack, we propose introducing zero-knowledge proofs in our protocol. Every user would have to prove that her outputs correspond to re-ordered and remasked versions of her inputs. This can be done using zero-knowledge proofs like the ones proposed in [Bras 87].

3. **Targeted public-key attack**: The key generation of Section 3.1 requires that all the users publish their shares at the same time. Otherwise, a participant can use the knowledge of the other shares to build her own. In this attack, a certain user U_j builds her key $y_j = g^{\alpha_j} / \prod_{i=1}^{n-1} y_i = g^{\alpha_j - \alpha_1 - \cdots - \alpha_{n-1}}$. Then, if y_j is used to build the group key, the result will be $y = g^{\alpha_j}$ and, hence, U_j will know the private group key.

   A solution for this attack is to broadcast previous commitments to the shares. Before sending her share, user U_i broadcasts a commitment $h_i = \mathcal{H}(y_i)$, where $\mathcal{H}$ is a one-way function. In the next step, the users exchange their shares and check that $h_j = \mathcal{H}(y_j)$ for $j = (1, \ldots, n)$.

The last modification can be efficiently integrated in our protocol. However, the introduction of zero-knowledge proofs causes the query delay to be, at least, twice longer. Introducing an unaffordable query delay while the security in front of the WSE remains the same is not acceptable for the protocol. In fact, we argue that the attacks conducted by internal users are not a real threat in practice. Note that, since the groups are created dynamically and randomly, there is a very small probability that an internal attacker falls into the same group as her victim twice. Furthermore, in order to build a complete profile of her victim, the attacker needs to join her in the same group several times. This requirement reduces even more the success probability of this kind of attacks.

4.4.2 Dishonest central node

The central node creates the groups of users. This entity only participates in the initial phase of the protocol, before the users exchange any message. Since it ignores any further communication between the users, the central node cannot link any query to the source. Therefore, the central node is not a threat for the privacy of the users.

4.4.3 Dishonest web search engine

The objective of the WSE is to gather the queries of the users in order to build their profiles. However, when the proposed protocol is executed, the WSE cannot know if a certain query has been generated by the user who has submitted it. This happens because when a user U executes the protocol, she submits her query hidden among other $n-1$ queries. Note that the other $n-1$ queries are not generated by a machine but by other real users. This means that the WSE cannot employ a method such as [Chow 09] or [Pedd 10] to distinguish human queries from automatically-generated ones. Therefore, the WSE can correctly select the query that belongs to U with a probability of $\frac{1}{n}$. Note that, in order to build a useful profile, it is not enough for the WSE to select correctly one query in one execution of the protocol. Since the probability of selecting always the correct query is very low, the result is that the WSE obtains a distorted profile of the user.

Nevertheless, this probability can increase if U submits several queries about the same subject. In each execution of the protocol, U hides her query among a group of n queries. These queries are generated by randomly chosen users and, hence, they are likely to be related to different subjects. As a result, after several executions, the WSE might be able to find the common subject in the queries submitted by U and build her profile. This situation can be avoided by periodically submitting queries about irrelevant subjects to U. TrackMeNot [Trac 13] follows a similar approach using machine-generated fake queries. [Chow 09, Pedd 10] prove that machine-generated queries can be distinguished from human queries. To solve this problem, the proposed scheme can use the different $n-1$ human queries which are gathered at each protocol execution. These queries can be stored in a database for later use.

4.5 Simulations

In order to evaluate the performance, the proposed protocol has been implemented and tested in a practical scenario. The objective of these tests is to prove that the query delay obtained by this protocol outperforms all the other proposals.

4.5.1 Implementation of the protocol

The implementation of the protocol is divided in two independent parts: the central node and the user application.

The central node is a process that is continuously listening to requests from the users. As stated before, when the central node receives n requests, a new group is created. The users of the new group receive a message with the information necessary to contact their partners. In order to enhance the performance of the protocol, this message also contains the large prime p, and the $g \in Z_p^*$ element. Consequently, the Step 1 described in Section 5.3.3 can be omitted.

The user application is a Java applet accessed by an html web page. Users employ this applet to type their query. Then, the protocol starts its execution. After that, the applet shows the corresponding results.

4.5.2 Time measures

Several simulations were performed in order to test the implementation of the protocol. The parameter selection was done using the same configurations proposed in [Cast 09]. This allows a later comparison between the UUP protocol and the new scheme.

In order to minimize the query delay, the creation of the group must be quick. According to [Cast 09], Google answers 1157 queries per second. The queries can be modeled using a Poisson distribution. This allows to calculate the probability of forming a group of n users in a certain amount of time. After several tests with $n = 3$, $n = 4$, $n = 5$ and $n = 10$, the authors of [Cast 09] conclude that $n = 3$ is the most realistic group size. As stated in [Cast 09], the probability of forming a group of $n = 3$ users in a hundredth of a second is close to 1.

Consequently, the group size for our simulations was $n = 3$ users, the key length used of the cryptographic operations was $l = 1024$ bits. The protocol was executed in an open environment (computers located at different places and connected through the Internet). Each computer executes a single user application in each simulation. The environment is the same as for the simulations of Figure 4.1.

The protocol was executed 1000 times by each user. The average delay measures obtained for these executions are shown in Figure 4.2.

The time intervals in the chart have been adjusted so that they can be compared with the time intervals of the UUP protocol. Since there are different steps in the new protocol, some time intervals disappear and some of them have been changed. Next, the meaning of each time interval is described:

- $t_0 - t_6$: these time intervals remain the same as in the UUP protocol.
- t_7: time interval required by the first member (U_1) to receive the ciphertexts c_i^0 from U_i for $i = 2 \dots n$.
- t_8: time interval required by U_i to *partially decipher*, re-mask and permute the received ciphertexts.
- t_9: time interval required by U_i to send the results which have been obtained after *partially deciphering*, re-masking and permuting the ciphertexts.
- t_{10}: time interval needed since the user U_i sends her ciphertext c_i^0 until the last user broadcasts the *decrypted* queries to all the group members. This period includes neither the time required to perform the partial deciphering, re-masking and permuting step (t_8) nor the time required to send the ciphertexts to the next user (t_9).
- $t_{11} - t_{13}$: the work done during these three intervals has been removed. In the new protocol, these steps no longer exist.
- t_{14}: time interval required by U_i to submit n queries to the WSE and receive the answers. Each user selects the answer that corresponds to her query.
- t_{15}: as stated in Section 4.2.1, this time interval (corresponding to the broadcast of the Google answers) is removed from the protocol.

4.5.3 Comparison between the two protocols

The UUP protocol obtained a query delay of 5.2 seconds [Cast 09]. As explained in Section 4.1.2, the size of the results page returned by Google has increased significantly in the last two years. Hence, nowadays, the expected query delay of this scheme would be 6.8 seconds approximately. The query delay achieved by the new protocol is 3.2 seconds. Therefore, the new proposal outperforms the results of [Cast 09]. It also outperforms the work presented in [Viej 10], which achieves the lowest delay (3.9 seconds) in the current literature. Note that this time was

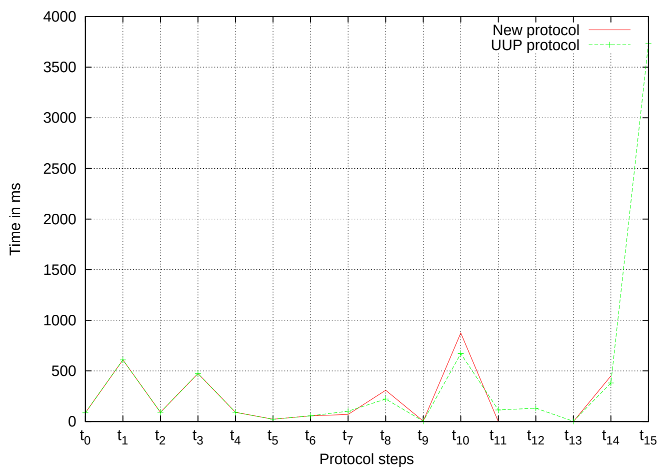


Figure 4.2: Partial times for the simulations made in an open environment with $n = 3$ users and $l = 1024$ bits.

calculated using a simulated scenario. Thus, [Viej 10] should be tested in an open environment in order to be properly compared with the new proposal.

Figure 4.2 shows the different delays obtained by the UUP protocol and the new scheme. Next, the main changes in partial times are remarked:

1. Intervals $t_0 - t_6$ are almost equal in both protocols. Note that, t_4 includes the use of a hash function to prevent chosen public key attacks.

2. Since the number of exchanged messages is lower, the delay of t_7 is reduced. During this interval, in the UUP protocol the ciphertexts were broadcast (*i.e.* all the members had to send and receive n messages). In the new protocol all the users send *one* message, while the first user is the only who receives n messages.

3. The introduction of the new operation called *partial decryption* affects the time intervals $t_8 - t_{13}$. Because of this operation, the computational time required by the operations performed during t_8 has augmented. In addition, the time that a user has to wait to obtain the queries in cleartext (t_{10}) has also slightly increased. On the other hand, the time intervals $t_{11} - t_{13}$ disappear in the new scheme.

4. In the UUP protocol users only submit *one* query to the WSE. In the new protocol they submit n queries. Figure 4.2 shows the delay caused by submitting

$n = 3$ queries. Note that, although this delay is higher than in the UUP protocol, it is not three times higher. This happens because the n queries can be submitted in parallel.

5. Figure 4.2 also shows that the UUP protocol has one step more than the new protocol. The longest time interval (t_{15} with more than 3.5 seconds) no longer appears in the new protocol. The removal of this delay is the most relevant improvement within the total time of the new scheme.

4.6 Conclusions

Web search engines have been proved to be a threat for the privacy of their users. Incidents like [M Ba 05] and [K Ha 06] show that users should not trust the companies behind the WSEs. Therefore, it is necessary to give to users a mechanism to prevent the WSEs from knowing their sensitive information.

Besides the need of privacy, another relevant issue for the users is the query delay. This is the time that users have to wait in order to obtain the search results for their queries. Current proposals increase significantly the query delay. This fact prevents these schemes from being successfully deployed in real environments.

In this chapter, we present a modification of the work presented in [Cast 09]. On one hand, our system optimizes some steps of the protocol to reduce the query delay. On the other hand, these changes incentivize every user to follow the protocol in order to protect their privacy. The new scheme has been tested in an open environment and the results show that it achieves the lowest query delay which has been reported in the literature.

Chapter 5

Distributed System for Private Web Search with Untrusted Partners

This chapter introduces a distributed protocol, where a group of users collaborate to protect their privacy in front of WSEs and dishonest users, while introducing a reasonable delay.

Contents

We introduce the novelty of the approach in §5.1, describing the security requirements to be guaranteed, the involved participants and the different phases. Section §5.2 describes the scenario and the requirements of the system. The protocol is detailed in Section §5.3. Section §5.4 analyzes the privacy of the protocol. Section §5.5 includes the implementation of the protocol, with the results of tests regarding its performance. Finally, Section §5.6 gives some conclusions and reports some future work.

5.1 Novelty of the approach

In this chapter, we present a multi-party protocol that protects the privacy of the users against web search engines and against dishonest internal users. Regarding similar approaches, we propose a protocol which increases the level of security of [Cast 09], and requires less computation and communication than [Lind 10]. More specifically, this work focuses on the second point of improvement (*level of security*) presented in Section 2.3.

Besides the description of the presented protocol, in order to analyze its behaviour, this chapter includes:

- The implementation of the protocol as a Java applet with a search engine interface.
- A test, using real data extracted from the AOL dataset, that evaluates the privacy of 1000 users, as if the had been executing the protocol during three months. In order to evaluate how their profiles change with the protocol, the test includes a morpho-syntactical and a semantical analysis of the submitted queries, and the application of an existing measure called the Profile Exposure Level (PEL) that estimates the privacy provided by the system.
- A group of tests, again employing real data from the AOL dataset, in order to estimate the time necessary to create a group. Results for different tests with different group sizes are presented.

- Performance tests where the protocol is executed in a controlled environment (i.e., a local network). Results for these tests show the delay of each interval of the protocol, aswell as the overall query delay (i.e., the total time that a user has to wait until she receives the query results). Moreover, these tests have been repeated for different parameter configurations in order to estimate which are the best values for the configurable parameters proposed in the protocol.

- Performance tests where the protocol is executed in an open environment (i.e., the Internet). The same points as in the controlled environenment have been analyzed. This test shows real time measures of the execution of the protocol via the Internet.

5.2 System Model

This section describes the entities that participate in the system. It also gives an overview of the proposed protocol and defines its requirements.

5.2.1 Entities

The protocol is executed in a scenario with three entities:

- *Users*. Individuals who submit queries to the WSE. We assume that in our scenario there are honest and dishonest users. The motivation of the honest users is to protect their own privacy. The motivation of the dishonest users is to learn the queries of the honest users.

- *Central node*. It organizes the users into groups. Its main objective is to distribute the information that users need in order to contact the other members of the group. This does not suppose a high workload and hence, we assume that the central node is run by one or more voluntary nodes. This assumption is also presumed by other systems like the Tor network [Ding 04], based on the use of a high number of voluntary nodes, and each one with a higher workload than the central node used in our proposal.

- *Web search engine*. It is the server that holds the database. We assume that it has no motivation to protect the privacy of their users.

5.2.2 Protocol Overview

The objective of the protocol is to create a group of users who collaborate in order to privately submit queries to a WSE. Instead of submitting her own query, a user U asks another member of the group to submit it and send the results back. At the same time, U submits the query of another user of the group and sends the results back to her. At the end, the WSE owns an obfuscated profile of each member of the group.

The protocol requires that neither the WSE nor the users of the group can link a query to a particular user. For this purpose, the users execute a distributed protocol that works as follows: a central node creates a group of n users, and provides them with the necessary information to contact each other. Then, the required OAS-Benes networks are built and fairly distributed among the n users. After that, each user encrypts and broadcasts her query. The list of encrypted queries is passed from each user to the next. In her turn, each user re-masks and permutates the list of ciphertexts at every switch that she was assigned. As a result, no user can link the inputs and the outputs of a particular user. Furthermore, for every switch, she uses PEP and DISPEP protocols to prove to the rest of users that the outputs are re-ordered and re-masked versions of the inputs.

The final result is that the users obtain a list of ciphertexts that cannot be linked to the original list. Then, each user decrypts one different query, submits it to the WSE and broadcasts the result.

5.2.3 Privacy and Performance Requirements

In order to guarantee the privacy of the users, the scheme must fulfill the following requirements:

- The users cannot link any query with the user who generated it.
- The central node cannot link any query with the user who generated it.
- The WSE cannot build a reliable profile of any user.

Regarding the performance, two parameters should be minimized:

- The time required to build a group of users.
- The delay caused by the communications and the cryptographic operations.

5.3 Protocol Description

The protocol is formed by four phases that users execute sequentially.

5.3.1 Group Setup

Every user who wants to submit a query to the WSE, sends a request to the central node. When the central node has received n requests, it creates a group $\{U_1, \ldots, U_n\}$. Then, the central node notifies the n users that they belong to the same group. The users receive a message with the size of the group (n) and the position that every component has been randomly assigned ($i = \{1, \ldots, n\}$). Each position is associated with the IP address and the port where the user is listening. This information allows the users to establish a communication channel between them. The central node is no longer needed and the rest of the protocol is P2P.

5.3.2 Permutation Network Distribution

As stated in section 3.3.1, t OAS-Benes networks are necessary to perform a secure permutation. The number of inputs of the networks equals the number of users $N = n$, which is also the same as the number of queries. Regarding the number of honest users, the parameter is always fixed at $\lambda = 2$. The reason for this choice requires a privacy analysis and, hence, it is later detailed in Section 5.5.4.

Knowing the parameters n, N, and λ, the users calculate the value of t using the formula defined on Section 3.3.1. The construction of the t OAS-Benes $PN^{(n)}$ is mechanical. This means that users do not need to exchange any information. As long as they know the parameters t and n, they know the arrangement of the switches in the t OAS-Benes $PN^{(n)}$. Therefore, they can fairly divide them into n adjacent stages.

According to the positions assigned in the previous phase, user U_i is responsible for the switches that correspond to the i-th stage. Each stage is formed by d switches, where $d = \frac{t}{n} \cdot S(n)$ on average.

We denote as s_l^i the l-th switch of the i-th user for $i = \{1, \ldots, n\}$ and $l = \{1, \ldots, d\}$. We also define a function $\Phi(i, l)$ that, given an output of a switch, it returns the input of the next switch that must follow. The result is given according to the arrangement of the switches in the PNs. Figure 5.1 illustrates the operation of this function.

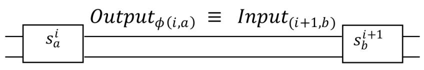


Figure 5.1: Correlation between outputs of a switch and inputs of the next

5.3.3 Group Key Generation.

1. Users $\{U_1, \ldots, U_n\}$ agree on a large prime p where $p = 2q + 1$ and q is a prime too. Next, they pick an element $g \in \mathbb{Z}_q^*$ of order q.

2. In order to generate the group key, each user U_i performs the following steps:

   (a) Generates a random number $a_i \in \mathbb{Z}_q^*$.

   (b) Calculates her own share $y_i = g^{a_i} \bmod p$.

   (c) Broadcasts a commitment to her share $h_i = \mathcal{H}(y_i)$, where $\mathcal{H}$ is a one-way function.

   (d) After receiving all the h_j for $j = \{1, \ldots, n\}$, $j \neq i$, she broadcasts y_i.

   (e) Checks $h_j = \mathcal{H}(y_j)$ for $j = \{1, \ldots, n\}$, $j \neq i$.

   (f) Calculates the group key using the received shares, as Equation 5.1 indicates.

$$y = \prod_{1 \leq j \leq n} y_j = g^{a_1} \cdot g^{a_2} \cdot \ldots \cdot g^{a_n} \tag{5.1}$$

5.3.4 Anonymous Query Retrieval

For $i = \{1, \ldots, n\}$, each user U_i performs the following operations:

1. U_i generates a random value r_i and uses the group key y to encrypt her query m_i, as Equation 5.1 indicates.

$$E_y(m_i, r_i) = (c1_i, c2_i) = c_i^0 \tag{5.2}$$

2. U_i sends c_i^0 to the other members U_j, for $\forall j \neq i$.

3. For every switch s_l^i ($l = \{1, \ldots, d\}$) with two inputs denoted as c_{i-1}^{2l-1} and c_{i-1}^{2l} received from U_{i-1} (note that the inputs for the switches of U_1 are the initial ciphertexts $\{c_1^0, \ldots, c_n^0\}$):

(a) U_i re-masks the cryptograms c_{i-1}^{2l-1} and c_{i-1}^{2l}. She obtains a re-encrypted version e_{i-1}^{2l-1} and e_{i-1}^{2l} using the re-masking algorithm defined in section 3.2.

(b) U_i randomly chooses $b_{i,l} \in \{0,1\}$ to determine the state of the switch s_l^i as in Figure 3.1(a). According to this state, she obtains a re-ordered version of the ciphertexts $e_{i-1}^{\pi(2l-1)}$ and $e_{i-1}^{\pi(2l)}$.

(c) U_i broadcasts $\{c_{\Phi(i,2l-1)}, c_{\Phi(i,2l)}\} = \{e_{i-1}^{\pi(2l-1)}, e_{i-1}^{\pi(2l)}\}$

(d) Assuming:

$$c_{i-1}^{2l-1} = E_y(m_1, r_1), \;\; c_{i-1}^{2l} = E_y(m_2, r_2)$$

$$e_{i-1}^{\pi(2l-1)} = E_y(m'_1, r'_1), \;\; e_{i-1}^{\pi(2l)} = E_y(m'_2, r'_2)$$

U_i must demonstrate that $e_{i-1}^{\pi(2l-1)}$ and $e_{i-1}^{\pi(2l)}$ are re-masked and re-ordered versions of c_{i-1}^{2l-1} and c_{i-1}^{2l}. This is equivalent to proving two statements:

I. $(m_2 = m'_2) \vee (m_2 = m'_1)$.
This can be proved using the DISPEP protocol of Section 3.5.

II. $(m_1 \cdot m_2 = m'_1 \cdot m'_2)$.
U_i computes $c = E_y(m_1 \cdot m_2, r_1 + r_2)$ and $c' = E_y(m'_1 \cdot m'_2, r'_1 + r'_2)$, and uses the PEP protocol (Section 3.4) to prove that c and c' are plaintext equivalent.

All the other users U_j ($\forall j \neq i$) verify the proofs.

4. Let us denote $\{c_1, \ldots, c_n\}$ as the resulting list of re-masked and re-ordered ciphertexts. At this point, each user U_i owns those n values. Then, user U_i decrypts the value c_i that corresponds to a query m^i generated by one of the group members. Note that due to the re-masking and permutation steps, m^i does not probably correspond to m_i (the query that has been generated by U_i).

   Decryption of a certain c_i requires that all n users participate by sending their corresponding shares to user U_i. According to that, U_i receives $(c1_i)^{\alpha_j}$ from U_j, for $j = (1, \ldots, n)$ and $j \neq i$. Then, U_i computes her own share $(c1_i)^{\alpha_i}$. Finally, U_i retrieves m^i as Equation 5.3 indicates.

$$m^i = \frac{c2_i}{c1_i^{\alpha_i}(\prod_{j \neq i} c1_i^{\alpha_j})} \tag{5.3}$$

5.3.5 Query submission and retrieval

At the last phase of the protocol, users receive the results for the queries that they generated. For this purpose:

1. U_i submits the retrieved m^i to the WSE.
2. U_i receives the results from the WSE and broadcasts them to the other members of the group.
3. At this point, each user has received $n-1$ results, and hence, she owns all the n results. From these n results, each user selects the ones that correspond to her original query and discards the rest.

5.4 Privacy Evaluation

This section analyzes the behaviour of the protocol regarding the privacy requirements defined in Section 6.2.1. The system is analyzed in the presence of the three dishonest entities that may participate in the protocol: dishonest user, dishonest central node and dishonest web search engine.

5.4.1 Dishonest User

The ElGamal cryptosystem is semantically secure under the Decisional Diffie-Hellman assumption [ElGa 85]. This means that a dishonest user cannot know if two different ciphertexts will result into the same cleartext after decryption. Therefore, every time that a ciphertext c_i crosses a switch, it is re-masked and permutated, and the attacker can only link the result to c_i by guessing, with probability of success $1/2$. This probability exponentially decreases for every switch that the ciphertext crosses. In the case of an attacker that only knows the inputs and the final outputs, the intermediate re-maskings and permutations prevent her from finding the links between them. Hence, given a particular user, the probability of correctly linking her with a decrypted query is $1/n$.

Let us consider the case where a dishonest user successfully learns the query of another component of the group. This means that she is able to link one input of the permutation networks with one of the outputs. This attack may be conducted if one of the following conditions is fulfilled.

1. *The dishonest user knows the secret group key*. The attacker can decrypt a query at any step of the protocol.

2. *The dishonest user ignores the key but knows the overall permutation*. Here, the attacker waits until the ciphertexts are decrypted. Then, she can link every query with the original ciphertexts and, hence, with their sources.

Regarding the first condition, the attacker can only recover the secret key if she compromises the $n-1$ other members of the group. The generation of the group key is distributed among the participants using the n-out-of-n threshold ElGamal key generation explained in Section 3.1.1. One of the characteristics of this scheme is that, if there is even a single honest user, the secret key cannot be reconstructed.

Another alternative in order to learn the secret key is to maliciously alter the key generation phase. In this phase, each user generates her share $y_i = g^{a_i}$. She then broadcasts a commitment to that share using a cryptographic function $\mathcal{H}(y_i)$, and she then sends y_i in a new message. A dishonest user may change her choice of share after receiving the shares of the other participants, before sending her own. This dishonest user calculates her share $y'_j = g^{a_j} / \prod_{i=1}^{n-1} y_i = g^{a_j - a_1 - \cdots - a_{n-1}}$ and broadcasts it. As a result, the group key is computed as $y = g^{a_j}$ and, hence, the dishonest user knows the secret group key. In order for this attack to be successful and remain undetected, the dishonest user must be able to find collisions in the hash function. This means that she must find a value y'_j for which her previous commitment is still valid (*i.e.*, $\mathcal{H}(y_i) = \mathcal{H}(y'_i)$). Nowadays, the probability of finding a collision in a reasonable amount of time, using a cryptographic hash function such as SHA-2, is almost negligible.

Regarding the second condition, the use of OAS-Benes PNs guarantees that the permutation is random and private. The requirement for this is that there must be at least one permutation network controlled by honest users. This means that the quantity of PNs that the scheme needs depends on the minimum number of honest users required to run the protocol. More specifically, the quantity of PNs that the scheme needs is the number that satisfies the following condition: in any possible distribution of stages among the users, the amount of switches controlled by the t honest users equals, at least, the number of switches composing one OAS-Benes PN. If this requirement is fulfilled, according to [Soo 02], the permutation is secure and remains secret to all the participants. Then, it is not possible to backtrace a permutation to find the original input.

5.4.2 Dishonest Central Node

This entity only participates in the initial phase of the protocol, before the users exchange any message. Since it ignores any further communication between the users, the central node cannot link any query to the source.

However, let us consider the case where a central node is in control of at least $n-1$ machines. Then, this entity could group a single honest user with $n-1$ users in its control. In this case, even if the protocol is thoroughly followed, the privacy of the honest user is lost. This happens because, at the end of the protocol, the queries are revealed and the central node can identify which query belongs to the honest user. Similarly, an attacker could send many requests to the central node so that it is likely that she controls a large fraction of the group. This situation can be prevented using a joint coin tossing scheme that uniformly distributes the parties controlled by the central node among all the groups executing the protocol. For example, the joint coin tossing scheme proposed in [Lind 10] can be fully integrated in our protocol. However, it is worth to mention that this solution has two shortcomings: (i) it requires that the number of peers controlled by the central node is small in comparison with the number of users ready to execute the protocol at a certain time; and (ii) executing the joint coin tossing scheme may be expensive.

5.4.3 Dishonest Web Search Engine

The objective of the WSE is to gather the queries of the users in order to build their profiles. In the proposed protocol, the WSE only participates in the last phase where it receives the queries from all the members of the group and returns the results. Therefore, the WSE can link each query with the user who submitted it and can include that information on her profile. Since a user U_i does not always submit her own query but also queries of other participants, after several executions of the protocol, the profile of U_i that the WSE owns is obfuscated.

In order to evaluate the level of privacy obtained by the users of the protocol, the following test is performed: let us assume that a user that submits her queries directly to the WSE (without using the proposed protocol) has a profile P. If she executes the protocol, we denote her resulting profile P' (the obfuscated profile). Therefore, in order to evaluate the level of privacy provided by the protocol, this test measures the difference between P and P'.

The first step in order to perform this test is to extract a profile from a list of queries. Several works like [Eick 13] and [Bals 12] characterize a profile as a set of categories that represent the topics of the user's submitted queries. For example, if a user submitted the queries "rheumatoid arthritis" and "basketball drills", her profile would contain categories "Health" and "Sports".

However, in practice, processing textual data and extracting their semantics is a challenging task. As stated in [Sanc 13], some linguistic analysis preprocessing must be applied. For our test, we have employed the method proposed in [Sanc 13]. This scheme has two steps:

1. A *morpho-syntactical analysis* over each query in order to obtain the main topic of the query. In this step, several Natural Language Processing (NLP) techniques based on *maximum entropy models* are used to syntactically analyze queries. These techiques are: sentence detection, tokenization, part-of-speech tagging, syntactic parsing, stop word removal, and stemming. A detailed explanation of each technique can be found in [Mann 99]. As a result of applying these techniques, we obtain the term (composed by a word or a set of words) that best represents the query topic.

2. A *semantical analysis* of the term obtained in the previous step using a knowledge base. For this purpose, the Open Directory Project (ODP) is used. ODP [ODP 13] is the largest human-edited directory of the Web, constructed and maintained by a large community of volunteers. The purpose of the ODP is to list and categorize web sites and for this, manually created categories are taxonomically structured and associated with web resources. Regarding semantical analysis, this structure can be exploited to interpret the meaning of a concept by finding the term in the structure and retrieving the upper categories of the taxonomy.

The result of applying this two steps is a list of ODP categories that classify the original query at different degrees. An example of this kind of classification is represented in Table 5.1. For instance, the query "how elvis prestley died" is found in the ODP ontology under "Arts : Music : Bands and Artists : P : Presley, Elvis". This means that its category *at first degree* is "Arts", *at second degree* is "Music", etc.

We use this method to transform the queries that the user intended to submit into P, and the queries that the user finally submitted into P'. Once profiles are categorized, some metric that quantify the difference between P and P' is

Query	ODP classification
how elvis prestley died	Arts : Music : Bands and Artists : P : Presley, Elvis
barbie doll houses	Recreation : Models : Dollhouse Miniatures
sexy swimwear	Shopping : Clothing : Swimwear : Women's : Bikinis
stamp framing	Recreation : Collecting : Stamps
weather	News : Weather

Table 5.1: Example of query processing to ODP categories.

needed. For this purpose, several approaches exist in the literature, among which we highlight [Bals 12], [Rebo 10], and [Erol 11b].

The first of them, [Bals 12], defines the average amount of information that the obfuscated profile reveals about the real profile as $H(P) - H(P|P')$. Where $H(P)$ is the entropy of P, and represents the uncertainty on the real profile; and $H(P|P')$ is the conditional entropy, i.e., the uncertainty on the real profile when the obfuscated profile is known.

The second of them, [Rebo 10], employs the Kullback-Leiber divergence [Cove 12], denoted as $D(P||\Lambda)$, where Λ represents an average user profile with the more popular interests. This measure is employed to evaluate the dissimilarity between the obfuscated and the average population profile.

The problem with both schemes is that they assume that the population profile is known. In [Bals 12], in order to calculate the entropy, it is assumed some background on the interests of the user population (e.g., which search topics are more popular). In [Rebo 10], the metric assumes that Λ, the average population profile, is known. However, for our tests, this assumption is difficult to accomplish, and a further analysis on the AOL dataset in order to obtain an approximate population profile would be necessary.

Consequently, and since it is not affected by the above assumption, we decided to use the third approach [Erol 11b]. In [Erol 11b], a metric called the Profile Exposure Level (PEL) [Erol 11b] is used. This metric measures the percentage of information about a user that can be extracted from her obfuscated profile. It is computed as equation 5.4 indicates.

$$PEL = \frac{I(\Phi, \Phi')}{H(\Phi)} \cdot 100 \tag{5.4}$$

Where Φ is the set of categories from the queries that the user intended to submit, Φ' is the set of categories from the queries that the user finally submitted,

$H(\Phi)$ is the entropy of Φ and $I(\Phi, \Phi')$ is the mutual information between both sets (i.e., if Φ' is known, how much does this reduce the uncertainty about Φ?). As a result, PEL measures the percentage of user information which is disseminated when Φ' is known.

Note that Equation 5.4 cannot be directly applied to all the categories in P and P' at the same time, but it is only useful to compare categories at a fixed degree. In the example of Table 5.1, comparing "barbie doll houses" with "stamp framing" at first degree ("Recreation" vs. "Recreation") computes as a maximum exposure, while comparing them at second level ("Models" vs. "Collecting") computes as a null exposure.

Regarding the way that the entropy and the mutual information are calculated in PEL, [Erol 11b] considers Φ and Φ' as two random discrete variables that have sample spaces Ω_Φ and $\Omega_{\Phi'}$ respectively. Then, the following probability functions are defined:

1. The probability function of Φ, denoted as $p(\varphi)$. It is defined as, $\forall \varphi \in \Omega_\Phi$, $p(\varphi) = P(\Phi = \varphi)$.
2. The probability function of Φ', denoted as $p(\varphi')$. It is defined as, for $\forall \varphi' \in \Omega_{\Phi'}$, $p(\varphi') = P(\Phi' = \varphi')$.
3. The conditional probability function of Φ given Φ', denoted as $p(\varphi/\varphi')$. It is defined as, $\forall \varphi \in \Omega_\Phi$ and $\forall \varphi' \in \Omega_{\Phi'}$, $p(\varphi/\varphi') = P(\Phi = \varphi/\Phi' = \varphi')$.

Using these probability functions, [Erol 11b] computes the entropy and the mutual information as Equations 5.5 and 5.6 indicate:

$$H(\Phi) = -\sum_{\varphi} p(\varphi) \cdot \log_2 p(\varphi) \tag{5.5}$$

$$I(\Phi, \Phi') = \sum_{\varphi, \varphi'} p(\varphi/\varphi') \cdot p(\varphi') \cdot \log_2 \frac{p(\varphi/\varphi')}{p(\varphi)} \tag{5.6}$$

In order to solve these equations, the probability functions must be defined. For this purpose, [Erol 11b] proposes the following notation:

$\Omega_\Phi = \{\varphi_i\}_{i=1}^{v}$, is the set containing the elements of Φ without repetitions.

$\Omega_{\Phi'} = \{\varphi'_i\}_{i=1}^{w}$, is the set containing the elements of Φ' without repetitions.

$Car_\Phi = \{car_{\varphi_i}\}_{i=1}^{v}$, is the set with the cardinals of each element of Φ.

$Car_{\Phi'} = \{car_{\varphi'_i}\}_{i=1}^{w}$, is the set with the cardinals of each element of Φ'.

$V = \sum_{i=1}^{v} car_{\varphi_i}$, number of elements of the set Φ, counting repetitions.

$W = \sum_{i=1}^{w} car_{\varphi'_i}$, number of elements of the set Φ', counting repetitions.

Finally, [Erol 11b] defines the calculation of the probabilities $p(\varphi)$ and $p(\varphi')$ as:

- Calculation of $p(\varphi)$ and $p(\varphi')$

  In this case, the probability of each element of Φ and Φ' is proportional to its cardinal. Consequently, $p(\varphi)$ and $p(\varphi')$ are computed as Equations 5.7 and 5.8 indicate.

$$P(\Phi = \varphi_i) = \frac{car_{\varphi_i}}{V},\ 1 \leq i \leq v. \tag{5.7}$$

$$P(\Phi' = \varphi'_i) = \frac{car_{\varphi'_i}}{W},\ 1 \leq i \leq w. \tag{5.8}$$

- Calculation of $p(\varphi/\varphi')$

  $P(\Phi = \varphi_i/\Phi' = \varphi'_j)$ is computed for each pair φ_i, φ'_j where $1 \leq i \leq v$ and $1 \leq j \leq w$.

  Fixing $\varphi'_j \in \Phi'$, two situations can arise:

  1. $\varphi'_j \notin \Phi$. There is no information, *i.e.*, the probability is randomly assigned among the elements of Φ and proportionally to their cardinal, as it is shown on Equation 5.9.

$$P(\Phi = \varphi_i/\Phi' = \varphi_j) = \frac{car_{\varphi_i}}{V},\ 1 \leq i \leq v. \tag{5.9}$$

  2. $\varphi'_j \in \Phi$. Then, there is a $\varphi_k \in \Phi$ so that $\varphi_k = \varphi'_j$.

     (a) If $car_{\varphi'_j} \leq car_{\varphi_k}$, it is assumed that φ'_j *corresponds* to φ_k (Equation 5.10, and not to any other $\varphi \in \Omega_\varphi$ (Equation 5.11).

$$P(\Phi = \varphi_k/\Phi' = \varphi'_j) = 1 \tag{5.10}$$

$$P(\Phi = \varphi_{k'}/\Phi' = \varphi'_j) = 0,\ 1 \leq k' \leq v, k \neq k' \tag{5.11}$$

(b) If $car_{\varphi'_j} > car_{\varphi_k}$, it is assumed that: (i) φ'_j *corresponds* to φ_k and not to any other $\varphi \in \Omega_\varphi$, with the probability proportional to the cardinal of φ_k, as Equation 5.12 shows; (ii) there is no information with probability proportional to the difference of the cardinals, as Equation 5.13 shows.

$$P(\Phi = \varphi_k/\Phi' = \varphi'_j) = \frac{car_{\varphi_k}}{car_{\varphi'_j}} + \frac{car_{\varphi'_j} - car_{\varphi_k}}{car_{\varphi'_j}} \cdot \frac{car_{\varphi_k}}{V} \tag{5.12}$$

$$P(\Phi = \varphi_{k'}/\Phi' = \varphi'_j) = \frac{car_{\varphi'_j} - car_{\varphi_k}}{car_{\varphi'_j}} \cdot \frac{car_{\varphi_{k'}}}{V},\ 1 \leq k' \leq v, k \neq k' \tag{5.13}$$

Parameter selection

In the proposed protocol, the privacy of the users in front of the WSE increases with the size of the group. This means that the larger the group is, the more privacy its members obtain. However, in practice, the size of the group is bounded by the time that users must wait in order to create the group. In order to minimize the query delay, the creation of the group must be as quick as possible. Simulations have been performed using three group sizes: $n = 3$, $n = 4$, and $n = 5$, considering that in each group, the n participants are different users. As explained in [Cast 09], the number of queries that Google answers per second can be modeled using a Poisson distribution. This allows to calculate the probability of forming a group of n users in a certain amount of time. After several tests, the authors of [Cast 09] conclude that $n = 3$ is the most realistic group size. However, in order to perform a more exhaustive analysis of our protocol, the protocol has also been tested for $n = 4$ and $n = 5$.

For the test, the AOL search logs have been used [M Ba 05]. Each entry of the logs contains five fields: an anonymous ID (*AnonID*), the query (*Query*) that she submitted, the time (day, hour, minute and second) at which the query was submitted (*QueryTime*) and, in some cases, the website that the user clicked (*ClickURL*) with the position that it filled in the results that AOL returned (*ItemRank*).

The test consists in simulating the execution of the protocol using groups of size $n = 3$, then $n = 4$, and finally $n = 5$, considering all the queries generated by a subset of 1,000 users during three months. In particular, we want to obtain the obfuscated profile that each user would have obtained for every group size, and compare it with her real profile from the AOL logs.

Results

The results obtained for this test are summarized in Table 5.2. The table contains the average PEL for different ODP categories. These percentages represent the amount of real profile information leaked by the observance of the obfuscated profile. As stated in [Bals 12], this is an indicator of the level of privacy provided by privacy-preserving protocols based on obfuscated profiles. In our case, it shows the difference between the uncertainty of the WSE before and after obtaining the obfuscated profile.

Table 5.2 shows the results for the first, second, third and fourth degrees of ODP categories. As argued in [Eick 13], ODP categories at the second degree privde a consistent, sufficient level of specificity in order to evaluate a profile. However, for the sake of exhaustivenes, we also present the results for the first, third and fourth degree. Additionally, as Table 5.1 shows, not all the queries are classified into the same number of categories. Consequently, the higher the degree of the ODP category, the less number of queries can be compared. For example, in our case 50% of the queries have less than five degrees. Applying PEL to a dataset where half of the data are missing is not very indicative of the profile exposure and hence, we decided to show the results only below the fifth degree.

Table 5.2: Average Profile Exposure Level at different degrees using groups of $n = 3, 4, 5$ users

	Degree 1	Degree 2	Degree 3	Degree 4
$n = 3$	67.27%	47.00%	40.73%	37.78%
$n = 4$	64.16%	41.03%	33.62%	29.43%
$n = 5$	63.91%	39.25%	30.84%	26.18%

The proposed protocol employs a permutation network (OAS Benes) that distributes the queries randomly between the n users that execute the protocol. For this reason, in theory, each user should submit her own query once in every n

executions ($1/n$ of times). Table 5.1 shows that the exposure level of P knowing P' is higher than 60% at first degree. This means that even if a user submits on average only $1/n$ of the queries that she generated, the submitted queries generated by other users share similar topics for the most general ODP category. At second degree the categories are more specific and hence, there is a greater diversity of topics, which lowers the profile exposure. The results for the third and fourth degrees show that the profile exposure level tends to $1/n$ for more specific ODP categories.

Consequently, from the results obtained for this test we can extract that, for general categories of ODP, the exposure level of P with regard to P' is high, while for specific categories of ODP, P and P' diverge. This means that executing the proposed protocol preserves the user's profile at general categories, but protects its specific contents.

5.5 Performance Evaluation

The performance evaluation is divided in two parts. First of all, we analyze the performance of our proposal comparing the results with the most similar proposals in the literature: [Cast 09] and [Lind 10]. Since the work presented in [Lind 10] does not include simulations nor a query delay estimation in a real environment, we decided to compare the protocols theoretically. Therefore, the two first tests are a theoretical analysis of the three protocols, evaluating the cost of computation time (*Test 1, Section 5.5.1*) and the number of messages that need to be exchanged in each case (*Test 2, Section 5.5.2*) .

Secondly, in order to complete the evaluation of our proposal, we have implemented the protocol of section 5.3. As stated previously, there is usually a tradeoff between the level of privacy that the user obtains and the performance of the protocol. For this reason, we have performed two different kinds of tests that evaluate the performance requirements defined in Section 6.2.1:

1. *Time to create a group (Test 3, Section 5.5.3) .* In distributed protocols where a group of users collaborate, the time required to create the group is a critical point. This period of time starts when the central node receives the first request from a user interested in executing the protocol, and ends when the central node has received n requests. Therefore, the third test aims at giving an approximate measure of the time required to build a group according to

the total number of users executing the protocol.

2. *Query delay (Tests 4 and 5, Section 5.5.4 and Section 5.5.5)* . The objective of the fourth and fifth tests is to evaluate the protocol in terms of performance, *i.e.,* the delay caused by the cryptographic operations and the messages sent through the network. In order to do this, the protocol has been implemented and tested in a controlled environment (Test 4) and a real environment (Test 5). The results of both tests allow to determine if the protocol introduces an affordable delay.

5.5.1 *Test 1*: Comparison of the Computation time

Next, we analyze the computation time needed in the execution of [Cast 09], [Lind 10] and our proposal.

Parameter selection of Test 1

Prior to the comparisons, three parameters of the system must be defined: the size of the group (n), the key length, and the number of OAS-Benes (t).

For the same reasons explained in 5.4.3, the group sizes are again $n = 3$, $n = 4$, and $n = 5$.

Regarding the key length, according to [Cast 09] and [NIST 07], a 1024-bit key length is considered computationally safe. In addition, the work presented in [M Ka 06] argues that a query is formed on average by 2.3 words and 15.5 characters. Assuming that a single Unicode character uses 2 bytes, a query would require 31 bytes on average. A key of 1024 bits can encrypt up to 128 bytes. This indicates that a system that employs a 1024-bit key length can accept queries with approximately 64 characters, a significantly higher value than the average query size.

The minimum number of OAS-Benes PNs, denoted as (t), is calculated according to the formula defined on Section 3.3.1. This formula depends on the size of the group (n), the number of inputs (N) and the minimum number of honest users (λ).

The selection of the size of the group (n) is explained above. The number of inputs equals the size of the group ($N = n$), because the inputs are the queries that every user generates. Nevertheless, the minimum number of honest users requires a further analysis.

Our scheme must be able to provide privacy in the worst possible conditions. That is, when the number of dishonest users is large in comparison with the number of honest users. However, the smaller the parameter λ is, the more OAS-Benes PNs are required and the higher the query delay grows. Hence, the value of λ must minimize the query delay without sacrificing the privacy of the users.

The minimum value for the number of honest users is $\lambda = 1$. However, this value does not guarantee the privacy of the users. As stated in Section 5.4.2, in a scenario with a single honest user and $n - 1$ dishonest users, even if the permutation is perfectly secure, the privacy of the honest user is lost. Note that a coalition of $n - 1$ dishonest users can easily identify which of the n queries belongs to the honest user.

The next possible minimum value is $\lambda = 2$. This value defines the worst case scenario in which our scheme can provide privacy. In this case, the $n - 2$ dishonest users have a probability of 0.5 of learning the query of the honest users.

In summary, we fix the parameter $\lambda = 2$ as the minimum number of honest users that our protocol requires. According to Formula 3.1, when $\lambda = 2$ the test for $n = 3$ requires $t = 2$ OAS-Benes PNs, while the tests for $n = 4$ and $n = 5$ require $t = 3$ OAS-Benes PNs.

Results of Test 1

As a result of this test, we show the amount of modular exponentiations that every user must perform in each execution of the protocol.

There are some parts of the protocol of [Lind 10] that employ a double encryption. This means that some modular exponentiations are performed modulus a 2048-bit integer value, instead of using a 1024-bit modulus like [Cast 09] and our proposal do. In order to compare the time required by a 1024-bit and a 2048-bit modular exponentiation, we executed a simulation that performed both operations. The simulation revealed that, in the same conditions, a 1024-bit modular exponentiation takes 22 ms on average, while a 2048-bit modular exponentiation takes 172 ms on average.

Table 5.3 shows the theoretical computation time needed by modular exponentiations in each protocol. The τ_{1024} denotes the time required to make one 1024-bit modular exponentiation. The τ_{2048} denotes the time required to make one 2048-bit modular exponentiation.

Figure 5.2 shows the calculated times for a group size of 3, 4 and 5 users. The

Table 5.3: Modular exponentiations average time for one user

Castellà et al. [Cast 09]	$(3n+3)\cdot\tau_{1024}$
Lindell et al. [Lind 10]	$6n\cdot\tau_{1024}+5n\cdot\tau_{2048}-\tau_{1024}+2\cdot\tau_{2048}$
Our Proposal	$\left(n+3+\frac{25\cdot t\cdot S(n)}{n}\right)\cdot\tau_{1024}$

results indicate that [Cast 09] obtains the lowest computation time. This happens because [Cast 09] does not use any mechanism to protect the participants against dishonest users. Since [Lind 10] uses double encryptions and our proposal uses zero-knowledge proofs, the computation times are higher. However, the results indicate that, regarding the modular exponentiations cost, our proposal outperforms the protocol of [Lind 10]. For example, for $n = 3$ users, our proposal requires approximately one second more of computation time than [Cast 09], while [Lind 10] needs 3 more seconds than [Cast 09].

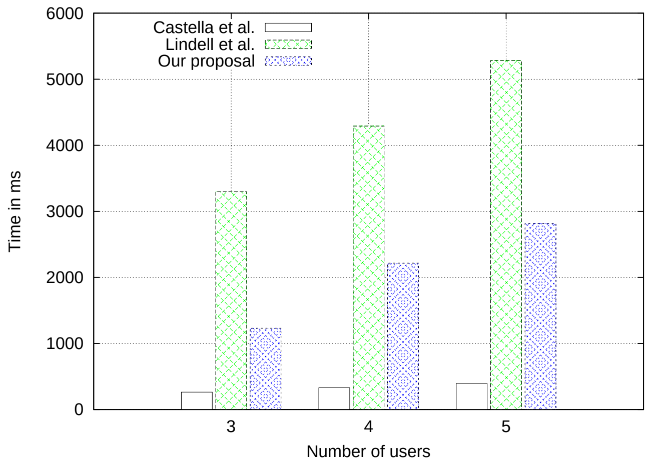


Figure 5.2: Comparison of modular exponentiations times per user

5.5.2 *Test 2*: Comparison of the number of messages

In order to analyze the performance of the protocol, another relevant information is the usage of the network. Table 5.5.2 reflects the number of messages that every user sends in each execution of the protocol. The parameters employed in this test are the same as for Test 1 (see Section 5.5.1).

Figure 5.3 represents the number of messages sent when 3, 4 or 5 users jointly

Table 5.4: Average number of messages sent by each user

Castellà et al. [Cast 09]	$3n - 1 - \frac{2}{n}$
Lindell et al. [Lind 10]	$4n - 2 - \frac{2}{n}$
Our Proposal	$4n - 4$

execute the protocol. Although the number of messages is similar in the three proposals, the results indicate that the number of messages sent in [Cast 09] is lower than in [Lind 10] and in our proposal. The results also indicate that our proposal requires less message deliveries than [Lind 10].

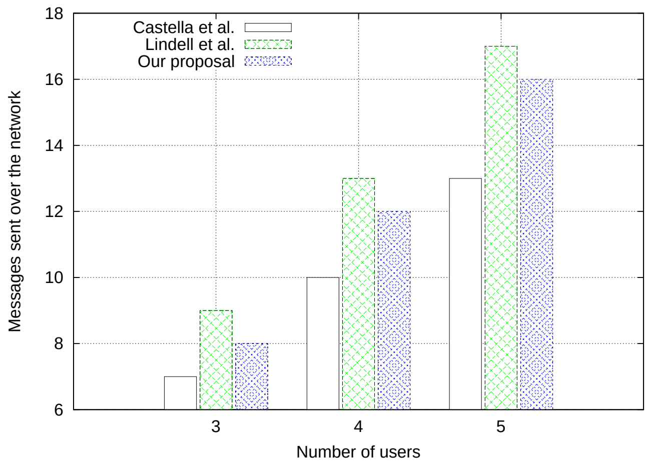


Figure 5.3: Comparison of messages sent in each protocol per user

5.5.3 *Test 3*: Time to create a group

In order to obtain this estimation, we have employed the AOL dataset [M Ba 05] again. As stated before, these logs contain a field called *QueryTime* that indicates the time at which each query was submitted. Therefore, using this field it is possible to simulate how the users of AOL would have been grouped if they had executed the protocol described in Section 5.3.

Parameter selection of Test 3

Again, as explained in 5.4.3, the group sizes are $n = 3$, $n = 4$, and $n = 5$. Therefore, the objective of *Test 3* is to determine the time to create a group of 3, 4, and 5 users for any total number of users employing the protocol. Instead of using the whole set of 658,000 users with their twenty million queries, we construct several samples or subsets of users. More specifically, we consider seven different samples. For example, the first sample contains the 2,784,908 queries belonging to 65,470 users during three months. Then, the rest of the samples are increasingly built. The $(i+1)$th sample contains all the users from the ith sample plus some other new users (always with their queries collected during three months). Consequently, the last sample contains the whole AOL dataset. Columns *Queries* and *Users* of Table 5.5 indicate the number of queries and users that each sample contains.

This division in samples allows to make a regression analysis. With this technique, we intend to estimate the relationship between two variables: the time required to create a group and the number of users employing the protocol. The result of this estimation is the regression function, which allows to predict the time to create groups of size $n = 3$, $n = 4$, and $n = 5$ for any amount of users that employ the protocol.

Results of Test 3

The resulting average times for each sample is shown in Table 5.5. Using these results, the regression function can be estimated as a potential regression. These estimation functions are shown in Table 5.6, where x is the number of queries generated by the users and y is the time required to create the group. The data obtained together with the estimation line of the regression is represented in Figure 5.4.

Table 5.5: Average time for creating a group of $n = 3, 4, 5$ users

			Delay		
Sample	*Queries*	*Users*	*n=3*	*n=4*	*n=5*
1	2,784,908	65,470	6395.96 ms	11536.23 ms	19402.73 ms
2	5,642,551	131,413	2942.95 ms	5087.46 ms	8579.86 ms
3	8,550,962	197,320	1891.75 ms	3192.97 ms	5094.81 ms
4	11,442,732	263,009	1399.07 ms	2220.07 ms	3524.80 ms
5	17,296,788	394,066	1017.90 ms	1420.04 ms	2537.52 ms
6	23,059,188	525,724	709.26 ms	942.17 ms	1854.25 ms
7	28,767,749	656,839	558.09 ms	558.10 ms	1556.03 ms

Group size	Regression function
$n = 3$	$y = 3 \cdot 10^{10} \cdot x^{-1.02}$
$n = 4$	$y = 10^{12} \cdot x^{-1.245}$
$n = 5$	$y = 2 \cdot 10^{11} \cdot x^{-1.086}$

Table 5.6: Regression functions for each group size.

The regression functions allow to estimate the time needed to create a group of $n = 3$, $n = 4$, and $n = 5$ users given any number of queries. For example, according to [Goog 12], Google answered 10,360,000,000 queries in a month. Therefore, in three months, Google would have answered 31,080,000,000 queries, approximately. Ideally assuming that all the queries were submitted using the proposed protocol, the regression function reveals that the time to create a group of $n = 3$ users would be 0.59 ms, for $n = 4$ users would be 0.66 ms, and for $n = 5$ users would be 0.81 ms. This means that in an ideal scenario where all the Google users execute the protocol, the time that each user must wait in order to find a group is insignificant.

5.5.4 *Test 4*: Query delay in a controlled environment

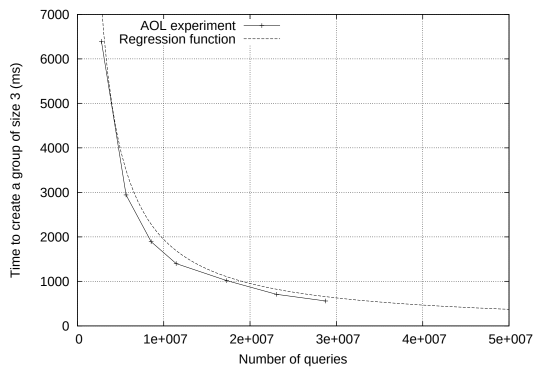
With the purpose of analyzing its performance, the proposed protocol has been implemented and tested in a controlled environment. The objective is to analyze the delay introduced by each phase of the protocol.

Implementation of the protocol

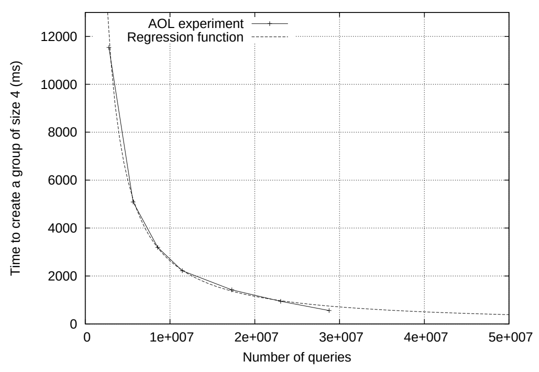
The implementation of the protocol has two independent components: the central node and the user application.

The central node is implemented as a process that continuously listens to requests from the users. According to the protocol description of Section 5.3, a new group is formed only after the central node has received n requests. Then, the central node sends to the users of the new group the information that they need in order to contact each other. For the sake of efficiency, the message that the central node sends also contains the large prime p, and the $g \in Z_p^*$ element. Thus, the Step 1 described in Section 5.3.3 can be omitted.

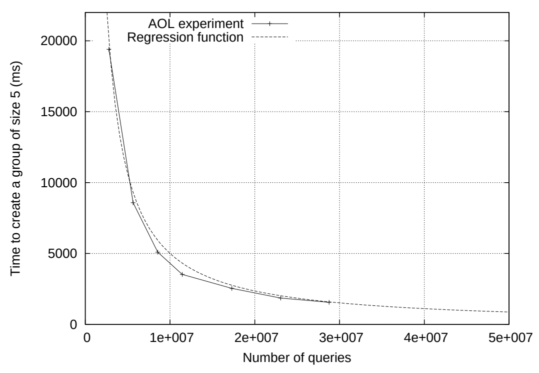
The user application is a Java applet with an interface similar to the interface of any WSE. Users employ this applet to type their query. The whole protocol is ex-



(a) Time to create a group of 3 users.



(b) Time to create a group of 4 users.



(c) Time to create a group of 5 users.

Figure 5.4: Time to create a group of size $n = 3$, $n = 4$, $n = 5$. Real results and estimation.

ecuted automatically and the user receives the results for her query transparently, as if she had directly employed the WSE.

Parameter selection of Test 4

Before executing the simulations, three parameters of the system must be defined: size of the group (n), key length, and number of OAS-Benes (t).

- *Size of the Group and Key Length.* Firstly, as in previous tests, the selected group sizes are $n = 3$, $n = 4$, and $n = 5$. Regarding the length of the keys employed to perform the cryptographic operations, we employ again a key length of 1024 bits, like in Tests 1 and 2. However, it is worth to mention that scenarios where the level of security required is critical usually use keys of 2048 bits. Therefore, for the sake of exhaustiveness, in this test we also present the results obtained when the protocol is executed with a key length of 2048 bits.

- *Minimum number of OAS-Benes PNs.* The minimum number of required OAS-Benes PNs (t) is calculated as for Test 1 and 2 ($t = 3$ OAS-Benes PNs).

- *Web search engine.* For the test, Google was selected as the web search engine where the users intend to submit their queries. Each user has a list of 1,000 randomly selected queries. The results presented for this test are the average of the values obtained when each user submits the 1,000 queries to Google using key lengths of 1024-bit and 2048-bit.

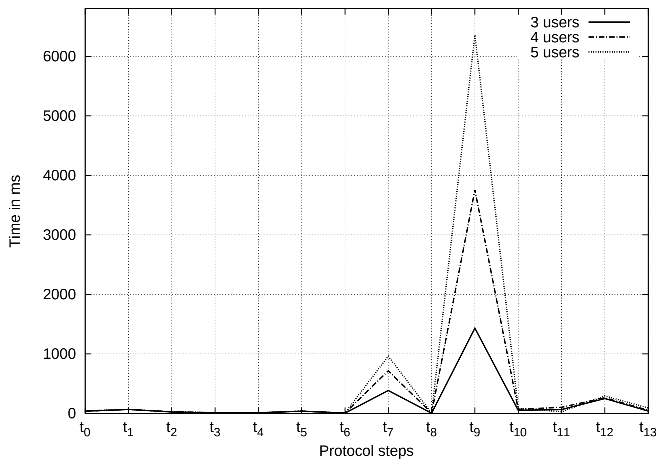
Equipment properties

The protocol has been tested in a controlled environment, where the users are connected using a Gigabit Ethernet wired local network. All the computers employed for the tests (including the central node) are *Intel Core i5 3.3 GHz* with 8GB of RAM. Each computer executes a single user application in each test.

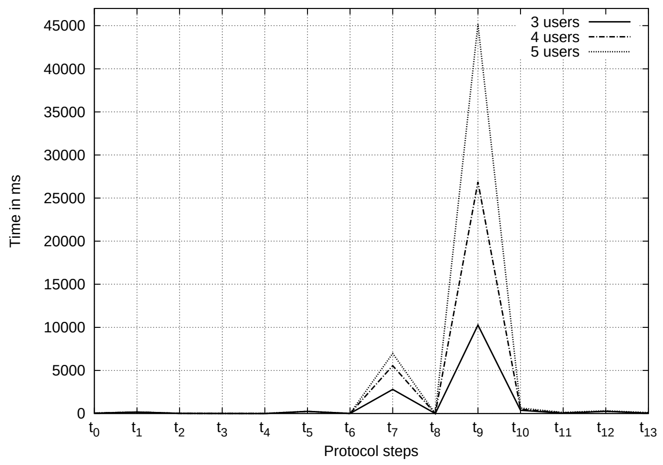
Time measures

In order to know which parts of the protocol have a higher delay, we have defined several time intervals that represent the delay of each protocol step. This is the list that contains the description of each time interval:

- t_0: time required by U_i to connect with the central node and to get a response. This response includes the information needed to contact with the other members of the group and the parameters to create the group key.



(a) Time intervals of the protocol using size groups $n = 3, 4, 5$ with a key length of 1024 bits in a controlled environment.



(b) Time intervals of the protocol using size groups $n = 3, 4, 5$ with a key length of 2048 bits in a controlled environment.

Figure 5.5: Protocol delays obtained in a controlled environment.

- t_1: time required by U_i to create her group key share α_i.
- t_2: time required by U_i to establish a connection with the other group members.
- t_3: time required by U_i to send her key share $y_i = g^{\alpha_i}$ to the group members.
- t_4: time required by U_i to generate the group public key y using the shares received from all group members.
- t_5: time required by U_i to encrypt the query m_i using the group public key

y.

- t_6: time required by U_i to send the resulting c_i^0.
- t_7: time required by U_i to pass the corresponding ciphertexts through the switches and generate their zero knowledge proofs (PEP and DISPEP).
- t_8: time required by U_i to send the results and the necessary elements of the zero knowledge proofs which have been obtained in the switches.
- t_9: time that U_i waits for the results of the switches of the other members and to verify the zero knowledge proofs.
- t_{10}: time required by U_i to calculate the shares which are used to decrypt the ciphertexts and send them to the group members.
- t_{11}: time required by U_i to decrypt the ciphertext c_i that she has received. Note that all the shares sent by the other group users are required.
- t_{12}: time required by the WSE to return the answer to the query m^i which has been sent by U_i.
- t_{13}: time required by U_i to broadcast the received answer.

Results of Test 4

For the test, the protocol has been executed with two key lengths (1024 and 2048 bits) and it has been repeated 1,000 times for each group size ($n = 3, 4, 5$). The graphic of Figure 5.5(a) and Figure 5.5(b) show the average values obtained for each time interval using a 1024-bit key and a 2048-bit key, respectively.

Results show that certain intervals introduce a significant delay to the system. For example, a high interval is t_7, i.e., passing the ciphertexts through the switches and generating the zero knowledge proofs. However, the highest time overhead is t_9. This is the time that users have to wait in order to receive the results of the other members and verify that the zero knowledge proofs are correct. From these two intervals (t_7 and t_9) two remarkable facts can be inferred:

- *The more the number of users, the higher the delay.* This happens because the number of switches required by an OAS-Benes PN grows with the number of users. When there are more switches in the OAS-Benes PN, there are more ZKPs to be generated and verified.

- *The higher the key length, the higher the operation cost.* There is a big difference between the operation cost using 1024-bit keys and 2048-bit keys. The cost of using a 2048-bit key is not twice the cost of a 1024 key-lenght. Results show that this cost grows exponentially.

Regarding the global result, Table 5.7 shows the delay that a complete single protocol execution requires on average. The table shows that the minimum delay (2.11 s, for 1024-bit keys and $n = 3$ users) is obviously higher than submitting the query directly to Google. This is caused by the overhead in the communications and in the cryptographic operations, and it is the downside for obtaining a higher level of privacy. Results also show a high delay for the executions with a key of 2048 bits (between 12 and 51 seconds, approximately). Therefore, the recommendation is to use the protocol with 2048-bit key length only in scenarios where the security is critical.

Table 5.7: Average time for executing the protocol with groups of $n = 3, 4, 5$ users

	$n = 3$	$n = 4$	$n = 5$
1024-bit key	2114.17 ms	4858.58 ms	8032.30 ms
2048-bit key	11920.22 ms	30497.85 ms	50918.57 ms

5.5.5 *Test 5*: Query delay in a real environment

With the purpose of comparing its performance in a controlled and real environment, the proposed protocol has been deployed in a real environment. The implementation of the protocol is the same as in Test 2.

Parameter selection of Test 5

As in Test 4, three parameters of the system must be defined: size of the group (n), key length, and number of OAS-Benes (t).

- *Size of the Group and Key Length.* Unlike previous tests, in this case the test has only been performed for a group size of $n = 3$ users. As previously mentioned, the authors of [Cast 09] conclude that $n = 3$ is the most realistic group size. Moreover, we argue that knowing the results for $n = 3$ in a real environment, and for $n = 3, 4, 5$ in a controlled environment (see Section 5.5.4), the results for $n = 4, 5$ in the real environment can be inferred.

Regarding the length of the keys employed to perform the cryptographic operations, the results are again presented for 1024 bits and 2048 bits key length.

- *Minimum number of OAS-Benes PNs.* As explained in Section 5.5.4, the number of OAS-Benes PNs for $n = 3$ is $t = 2$. .

- *Web search engine.* For this test, Google was again the selected search engine. As in Test 4, the results presented for Test 5 are the average of the values obtained when each user submits the 1,000 queries to Google using key lengths of 1024-bit and 2048-bit.

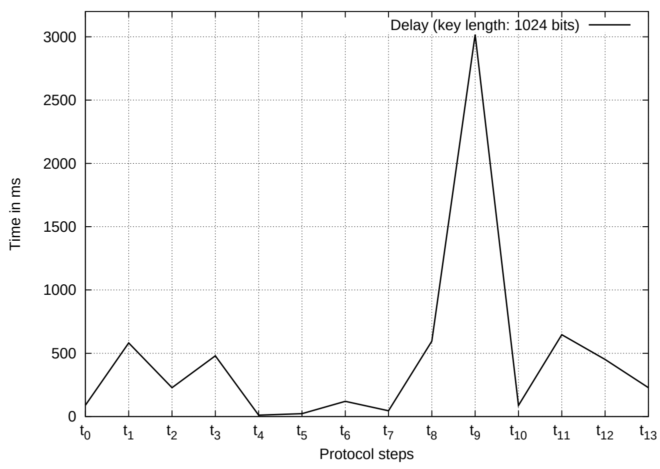
Equipment properties

The protocol has been tested in an open environment. The central node is executed in a server that is continuously listening to requests. The $n = 3$ computers that execute the user application are located in different cities and, therefore, connected between them through the Internet. Regarding the specifications of the computers involved in the tests: the server is an *Intel Pentium 2.8 GHz* with a *100Mb/100Mb* Internet connection. The other three machines are standard desktop computers with *Intel Core processors ranging from 2GHz to 2.7GHz* and Internet connections ranging from *3Mb to 10Mb* for download and *300Kb to 1Mb* for upload. Each computer executes only a single user application in each test.

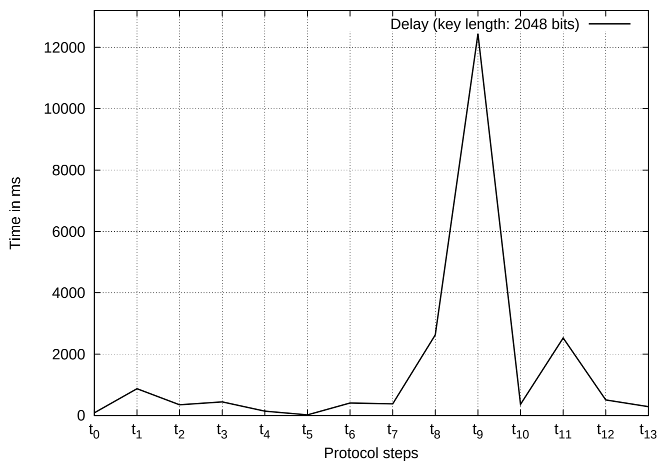
Results of Test 5

For the test, the protocol has been executed with two key lengths (1024 and 2048 bits) and it has been repeated 1,000 times for a group size $n = 3$. The graphic of Figure 5.6(a) shows the average values obtained for each time interval using a 1024-bit key. Note that the time intervals t_0 to t_{13} are the same time intervals defined for Test 4 (see Section 5.5.4).

The results show that certain intervals increase in comparison with the results obtained in the controlled environment. As opposed to the controlled environment, all the intervals in the real environment are affected by the network latency. For example, intervals t_0 and t_2 (the intervals where the users connect to the central node and between them) take approximately one second each one. Additionally, the intervals where the users exchange any message (t_3, t_6, t_8, t_9, t_{13}) are also significantly affected by the network latency.



(a) Time intervals of the protocol using a group of size $n = 3$ with a key length of 1024 bits in a real environment.



(b) Time intervals of the protocol using a group of size $n = 3$ with a key length of 2048 bits in a real environment.

Figure 5.6: Protocol delays obtained in an real environment.

However, the highest time overhead is t_9. This interval includes the time where the users have to wait in order to receive the results of the other members and verify that the zero knowledge proofs are correct. Because of the network latency users have to wait longer in order to receive the ZKPs and consequently, longer in order to complete the verification.

Regarding the overall delay, a single protocol execution requires 6.6 seconds using a 1024-bit key, and 21.4 seconds using a 2048-bit key. As previously mentioned, it is only recommended to employ 2048-bit keys in scenarios where the security is critical. For 1024-bit keys, although the cost is higher than directly submitting the query to the WSE, the obtained results are still affordable, specially since the protocol protects users not only in front of the WSE, but also in front of internal attackers. These results can be compared with those obtained by the most similar works in the literature, described in Chapter 2. The UUP protocol [Cast 09] obtains a 5.2 second delay for $n = 3$ users and 1024-bit keys, but is vulnerable in front of internal attackers. On the other hand, [Lind 10] is resilient against these attacks, but the analysis performed in Tests 1 and 2 indicates that it is 20 times slower than the UUP protocol. Our protocol is only 1.3 seconds slower than the UUP protocol and achieves the same level of privacy than the protocol presented in [Lind 10]. Accordingly, it can be inferred that our protocol obtains the best trade-off between the level of privacy and the time of response.

5.6 Conclusions

This chapter introduces a distributed protocol which protects the privacy of the users in front of WSEs and dishonest internal users. The proposed scheme has been compared to similar approaches in the literature. Results show that it outperforms proposals that provide the same privacy level. Additionally, the protocol has been implemented and evaluated using real queries from real users in a controlled and real environment. On one hand, the tests which have been performed give an estimation of the overhead caused by the new protocol when submitting a query. In its faster configuration, a single protocol execution in a real environment with $n = 3$ users and 1024-bit keys requires 6.6 seconds. On the other hand, they also give a measure of the level of privacy exposure that each user achieves executing the proposed protocol. More concretely, after computing the PEL measure, it can be inferred that executing the proposed protocol preserves the user's profile at a general level, but protects its specific contents.

Chapter 6

Design of a P2P network that protects users' privacy in front of Web Search Engines

This chapter introduces a P2P architecture that groups users according to their interests in order to protect their privacy in front of a WSEs

Contents

Section 6.1 introduces the main contributions of this chapter. Section 6.2 defines the requirements of the system. Section 6.3 explains the new scheme. Section 6.4 analyses the performance of the proposal. Finally, Section 6.6 outlines the main conclusions and future work.

6.1 Novelty of the approach

Chapter 5 states on Section 5.4.2 that a dishonest central node is a dangerous adversary and it can be considered a bottleneck. For this reason, in this chapter we present another privacy-preserving multi-party protocol where the central node is no longer needed.

More specifically, we introduce a P2P architecture that organizes users into *groups* according to their interests. Then, the architecture is used as an overlay, and inside each *group* a privacy preserving multi-party scheme like [Viej 10] is executed. We have selected this one because it is compatible with our architecture, and its application in the new scenario solves the shortcommings presented in Section 2. The overall objective is to achieve a good trade-off between the privacy level and the quality of the service, while offering robustness, scalability and well-balanced traffic over the network. Therefore, this chapter focuses on the third point of improvement (*quality of the retrieved results*) described in Section 2.3.

The advantages that the proposed scheme offers with respect to previous works are:

- *Human-generated queries*. Users inside the network submit queries generated by other real users. Hence, the scheme does not suffer from the problem of machine-generated queries detection that single-party schemes used to have.
- *Accurate group profile*. Users are organized into groups according to their interests. Within these groups, users execute a multi-party protocol like [Viej 10]. This ensures that users only submit queries from users that share the same interests. As a result, users obtain a *group profile* that WSEs can employ to improve users' experience (desambiguation, personalization of the results, suggestions, etc.).
- *Flexibility*. The protocols presented in [Viej 10] and [Erol 11b] are designed to be deployed in social networks. Both protocols assume that (i) users have

a social network account; (ii) users' computers are permanently connected to the Internet. Nevertheless, these assumptions may not always be true. The special-purpose network that we propose can be employed by any user without requiring a social netwok account. Additionally, the network manages the joins and leaves of users and hence, we do not require the users' computers to be permanently connected to the Internet.

- *Simulation tests with real data*. The proposed P2P architecture has been implemented using a simulator called PlanetSim [Ahul 08]. Then, the simulations have been performed using real data extracted from the AOL file [M Ba 05]. In this way, the correct behaviour of the proposed system has been tested with data generated by real users.

6.2 System requirements

In order to guarantee the proper behaviour of the system, two kinds of requirements are defined: privacy requirements and performance requirements.

6.2.1 Privacy requirements

Privacy requirements consist in protecting users against three kinds of adversaries:

- *Web search engine.* It is the main adversary in our scenario. Its objective is to learn the queries generated by each user in order to build a *detailed* profile.

- *Dishonest user.* A dishonest user profits her insider role in the protocol to learn the queries from other members. She knows which of her neighbours has forwarded a particular query to her.

- *Selfish user.* A selfish user does not follow the protocol. She utilizes the network in order to privately submit her own queries. However, she does not collaborate to submit the queries of the other members. Several users acting this way can cause a denial of service and prevent honest users from obtaining their search results.

Regarding the two first adversaries (WSE and dishonest user), the requirement is to prevent them from learning the source of any query. Regarding selfish users, the system is required to penalize this kind of behaviour.

6.2.2 Performance requirements

As for performance, the following requirements are defined in the system:

- *No bottleneck.* The system must be P2P, without a central node that creates and manages the groups.
- *Scalability.* The system must be able to work with a high number of users, maintaining an acceptable response time.
- *Robustness.* The system must support frequent joins and leaves of users, while preserving the connectivity between nodes (i.e., no user can remain isolated).
- *Accurate grouping.* The system must group users with similar profiles. If users within a group have too different profiles, they will submit queries that do no match their interests. This may result in a distorted group profile, which prevents WSEs from providing a good quality of the service.

6.3 Our proposal

In order to match the privacy and performance requirements defined in Section 6.2, we propose a P2P architecture used as an overlay to protect the users' privacy. This section describes the details of the proposed architecture.

6.3.1 Entities

There are three different entities that participate in the system: the peers, the superpeers, and the bootstrap.

- *Peers.* They are the users of the system. Their objective is to privately submit their queries to the WSE.
- *Superpeers.* They are peers that have additional functions. They are in charge of the organization of the network. Their function is to arrange the users of the system, redirecting them to their assigned place in the network and providing the information required to contact other users.
- *Bootstrap.* It is the input point of the system. It is a node or a set of nodes that redirects new users to the superpeers. The bootstrap is maintained by

the owners and/or volunteers of the P2P network. Its physical address is publicly available.

6.3.2 Architecture design

The three entities that participate in the system are connected inside the network in a three-layer architecture.

The first layer is composed by the bootstrap. The bootstrap can be a unique node or a set of nodes, depending on the size of the network. For a large network, a larger number of nodes may be necessary. Every bootstrap node contains a cache that stores the physical addresses of some superpeers. A bootstrap node must know at least one superpeer of the network, but it is not required to know all the superpeers.

The second layer is formed by the superpeers, which are connected between them using a P2P overlay. The characteristics of this overlay are later discussed in Section 6.3.2.

The third layer contains the peers, organized in structures called *Profile Clusters*. A Profile Cluster is an unstructured P2P network that contains peers that share similar interests. Each Profile Cluster is managed by a different superpeer. This superpeer holds an *input queue* that keeps the physical addresses of the peers in the Profile Cluster. A FIFO policy is applied in the queue: every time that a new peer joins, her address is included in the queue and the address of the oldest peer is removed.

Inside the Profile Cluster, users are not connected in a complete graph. Instead, each of them is connected to σ neighbours, where σ is a value that changes over time, when new peers join and others disconnect. However, for privacy reasons later described in Section 6.3.4, $\sigma \geq 2$, i.e., there is no peer that has only one neighbour.

The entities as well as the connections between them are represented in Figure 6.1.

The next two subsections give a detailed explanation of the second and the third layer, respectively.

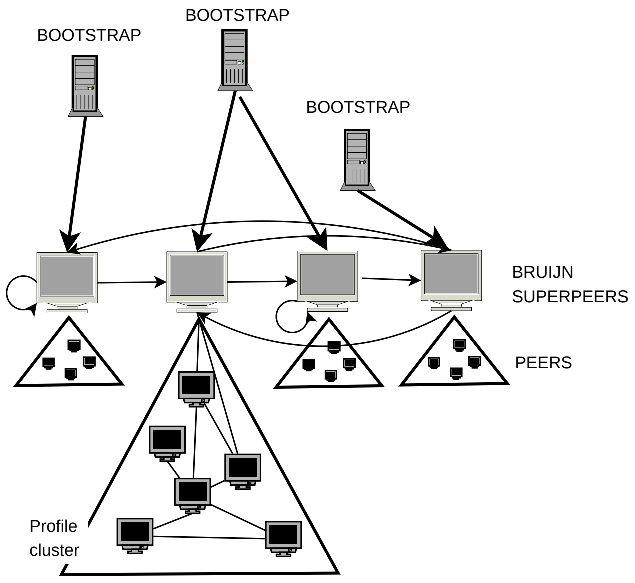


Figure 6.1: Architecture of the P2P network

P2P Overlay

Our objective is to group users according to their profile and to hide their real queries from the WSE. We propose the design of a P2P network of low latency that supports a huge amount and mobility of users.

In the literature, there are several proposals that provide scalable, self-organized P2P networks, offering different services such as distributed computing, message passing, document and file sharing, p2p games, etc. Next we describe the most recognized P2P overlays.

CAN [Ratn 01] is a topology which uses a multi-dimensional Cartesian coordinate space in form of torus. Each peer has a portion of the Cartesian space assigned and manages a table with information about neighbours (IP and virtual coordinates). CAN uses *greedy* routing strategy, where a message is routed to the neighbour that is *closer* to the required location. The CAN algorithm is highly scalable and self organizing.

Chord [Stoi 01] is a popular and common topology which organizes peers using a DHT (Distributed Hash Function) and assigns a long key id, originally 160 bits, to every node. The topology is organized as a circle where each node contains a successor and predecessor link. In addition, each node manages a neighbour table with a log_2N size, which contains the key and physical address of the 2^{i-1}

successor in the i_{th} position. Therefore, a node can reach any other node with a logarithmic cost.

Tapestry [Zhao 01] is based on the work of Plaxton et al. [Plax 97], which uses an internal structure of data in order to find the peers, correlating a node ID and a final key to route a message. It also implements multiple replicas for each object to prevent a unique point of failure. The routing map is composed by $log_B N$ levels. Each peer manages a routing table of B entries for each level. In the same way as Pastry, supports replication of data and track of multiple paths. A special feature of Tapestry is that it supports replicate data across multiple peers and keeps track of multiple paths to each peer to provide flexibility when sending messages (surrogate routing). It also addresses the problem of a single point of failure (when an object has a single peer) by assigning multiple roots to each object.

Pastry [Rows 01], similarly to Tapestry, implements a Plaxton mesh. However, in this case, its routing protocol consists in pairing the destination key with the actual prefix. In the same manner, the neighbour table manages $log_B N$ rows, but only stores $B-1$ entries. The maintained relation is that the row n stores the address of nodes which shares the n first digits of the key, but the $n+1$ and upper positions contain the other $B-1$ possible values according to its own key. Similarly to Tapestry, Pastry supports replication of data and track of multiple paths.

Kadmelia [Maym 02] implements a routing protocol based on XOR metric to obtain the distance between two points in the key space. The distance between two points is defined as $d(a,b) = a \oplus b$. In order to implement it, each peer has to manage a double linked list of k nodes, named $k-buckets$, where in each position i stores the most recent nodes between the 2^i and 2^{i+1} position with the same prefix. The value of k is usually 20. XOR is symmetric and therefore, it allows peers to send and receive queries from the peers of their routing tables. Furthermore, a peer can send a query to a peer within an interval, while allowing to select routes based on latency or parallel asynchronous queries.

Viceroy [Malk 02] manages the scheduling and localization of data and resources in an approximation to a *Butterfly* network. It applies a uniform DHT function to distribute the set of data. The topology is arranged in $log_2 N$ levels, where each level is a ring that contains an even number of peers connected by means of $level-ring$ links. Note that these peers are chosen to long range contacts. On the other hand, the interconnection of levels is made up by using $Down - Left$ and $Down - Right$ edges, and they are used to perform short contacts. This topology stands out from the rest because it only requires a constant number of links, 7 for each peer.

Bruijn [Logu 03, Kaas 03] is a graph of logarithmic diameter $log_k N$ used as an overlay. Each peer contains k incoming and outcomming links. The build of the links H of each peer consists in producing a shift operation of 1 bit to the left. The final result is a directed *graph* where the links are distributed symmetrically around the network. It has a constant degree, such as Viceroy, but Bruijn is more flexible because it allows to change the number of connections according to the specified base k.

N-ari Tree [Cast 11b, Jaga 05] is a hierarchical topology in tree form. The topology is based on a *N-Ari Balanced Tree* structure. It is built starting from the root peer, where it is connected to B peer nodes, known as *son* peers, and which are situated in the immediate lower level. Simultaneously, these nodes can be the parent nodes from other *son* peers. The final nodes that do not have any *son* peers are known as leaves. If the tree is balanced, the number of levels in terms of number of peers reaches $log_B(N(B-1)+1)-1$ and achieves a constant degree of nodes.

overlay	Degree	Diameter
CAN	$2 \cdot d$	$d \cdot N^{(\frac{1}{d})}$
Chord	$\log_2 N$	$\log_2 N$
Tapestry	$k \log_k N$	$\log_k N$
Pastry	$(2 \cdot k) \log_k N$	$\log_k N$
Kadmelia	$k \cdot \log_k N + k$	$\log_k N + c$
Bruijn	k	$\log_k N$
Viceroy	7	$\log_2 N$
Baton (N-ari tree)	$\log_2 N$	$2 \log_2 N$

Table 6.1: Degree-diameter complexities.

Our contribution tries to take advantage of the P2P overlay and the methods for addressing multi-attribute capabilities of the peers, extracting the computing trends and redesigning some levels for classifying and optimizing according to the WSE profile. With this objective, a two-level overlay is proposed. At the first level *Bruijn* was chosen from among the most widely used P2P topologies [Mesh 08, Lua 05] to form the base overlay for connecting the *superpeers*, mainly

overlay/k	2	3	4	10	20
CAN	1414	144	45	5	3
Chord	20	-	-	-	-
Tapestry	20	13	10	6	5
Pastry	20	13	10	6	5
Kadmelia	23	15	11	7	5
Bruijn	20	13	10	6	5
Viceroy	31	20	16	10	8
Baton (N-ari tree)	-	40	26	13	10

Table 6.2: Diameter ($N = 10^6$ nodes).

because it offers the best diameter-degree. Bruijn is employed as a structured network with the capability of finding attributes efficiently. It provides the performance requirements of the underlying framework outlined in Section 6.2.2, that is, *no bottleneck*, *scalability*, *dynamism* and robustness [Kaas 03]. Table 6.1 shows the degree-diameter values of the different structured networks discussed earlier. The high connectivity of Bruijn, with the best pair degree-diameter can be appreciated. Table 6.2 shows the growth of diameter-overlay for a maximum number of nodes $N = 10^6$. This confirms also the good scalability of Bruijn.

The second level of our overlay is an unstructured network, *Profile cluster*. The Profile Cluster groups peers that share similar interests used to implement the privacy protocol. As it is defined in literature, the unstructured networks are easier to build because the arrangement of nodes arbitrary. Structured networks (Can, Chord, Tapestry, Pastry, Kadmelia, Viceroy, Bruijn and N-ari tree) were discarded because they are not suitable for privacy reasons. As later explained, for the sake of privacy, it is necessary that the network topology remains unknown. Therefore, privacy between nodes is better achieved with decentralised and non-structured overlays.

In previous works, Castella et al. [Cast 10, Cast 11a] proposed a multi-layer P2P computing overlay (DisCoP) optimized for searching multi-attribute computational resources, such as CPU, memory and secondary storage, in a completely distributed and scalable manner. We adapted the idea of DisCoP of having a multi-layer scheme. We consider that two layers are enough to fulfill our requiremens, instead of the three layers proposed by the authors of DisCoP.

Profile Clusters

In order to arrange the peers into a Profile Cluster (the second layer of the P2P overlay) it is necessary to define their profiles. For this purpose, each user owns a vector that represents her interests (the Profile Vector).

Consider k categories of interests such as *news*, *sports*, *shopping*, *science*, etc. Then, the *Profile Vector* of a peer, denoted as V, contains k coordinates:

$$V = \{C_1, .., C_j, .., C_k\}$$

Each component C_j indicates the level of interest, ranged between $[0..1]$, in the j^{th} category. For design reasons, the number of bits of each C_j is equal.

In order to define the k categories employed in the system, the Open Directory Project (ODP) [ODP 13] is used. The ODP is an ontology that organizes the World Wide Web links. It has a hierarchical structure, where the topics are arranged in categories which in turn can also include smaller categories. For example, for the category *shopping*, some subcategories exist. These can be *clothing*, *health*, *holidays*, etc.

Due to the large size of ODP, our system only considers the most relevant k categories of the first level. However, including subcategories in the Profile Vector V is possible. In doing so, the corresponding bits assigned to encode each C_j are divided by the number of subcategories. Then, the significance of each subcategory is weighted and set in one part of C_j. Thus, the format of Profile Vector V remains intact.

6.3.3 Building the system

Similarly to most P2P systems, the network is built dynamically and ad-hoc. As new nodes attempt to enter the system, they are gradually connected to the existing ones. There are two main steps to insert a new peer P in the system:

1. First of all, it is necessary to categorize the Profile Vector of P. For this purpose, the user selects her level of interest in each category C_i defined in the system. As a result, the user obtains her Profile Vector V. Then H, the Hilbert transform of V is obtained. This is $H = Hilbert(V)$. A Hilbert curve [Moon 96] is a Multi-dimensional Space Filling Curve (SFC) overlay. A Hilbert transform converts a multi-attribute vector (V) into a uni-

dimensional vector (H). Unlike other SFC curves, Hilbert assures that the locality between objects in the multi-dimensional space is also preserved in the linear space. This feature even increases the classification of peer profiles. We use this to group peers into Profile clusters, the second level of our P2P overlay.

2. P contacts Bootstrap and receives the physical address of a superpeer SP_a in the Bruijn network. SP_a is chosen randomly from the list of current superpeers. Three situations can arise next:

   (a) There is no Superpeer in the list of current superpeers. Then, P becomes the initial superpeer and joins the Bruijn network, as Section 6.3.3 describes.

   (b) Superpeer SP_a asks to the other superpeers in the first network level (Bruijn) if a Profile Cluster associated to H exists. Two situations are possible:

      i. $\nexists$ Profile Cluster for H. Similarly to the previous case, P becomes a new superpeer of the Bruijn network (see Section 6.3.3).

      ii. $\exists$ Profile Cluster for H. SP_a redirects P to the superpeer SP_b in charge of the Profile Cluster that best matches its Profile Vector V. In this case, P is inserted as a regular peer inside the SP_b Profile Cluster. In order to bind to its Profile Cluster's neighbours, it executes the procedure detailed in Section 6.3.3.

Building the Bruijn network

The first level of the P2P overlay, Bruijn, is formed by Superpeers.

A Bruijn graph is characterized by two parameters: (a) the degree k or number of input and output links of each node (the degree k also designates the numerical basis of the key assigned to each node) and (b) the diameter d. This is the maximum distance between any pair of nodes (equivalent to the length of the key nodes). Thus, the maximum number of nodes in the graph is $N_{Bruijn} = k^d$. Following this notation, the nomenclature Bruijn(k,d) is used to describe a graph of a particular size. Fig. 6.2 shows an example of a complete graph Bruijn(2,3) with $N_{Bruijn} = 8$ nodes.

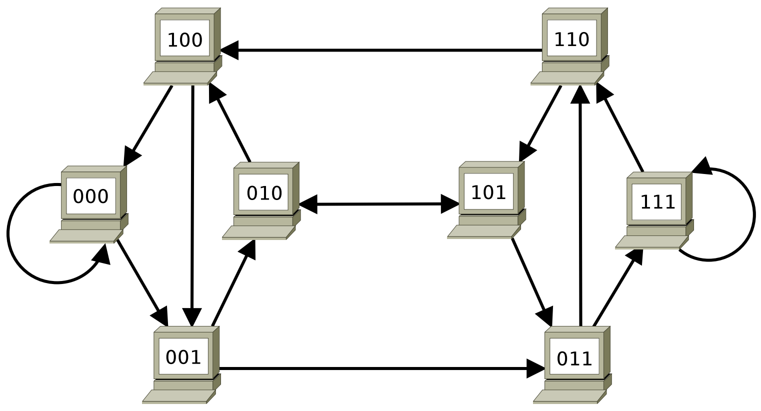


Figure 6.2: Directed graph Bruijn(2,3) with $N_{Bruijn} = 8$ nodes.

Bruijn graphs are uniform, multi-directional and symmetric. These properties allow the message traffic to be balanced across the topology, avoiding congestion and bottlenecks. Another advantage of the Bruijn is that a node only has to maintain a small and constant number of links, which imples a low construction cost for the topology and which also means that it is more robust.

Furthermore, in our case, each node has a successor and a predecessor (as a ring topology). The successor of a node is the closer node in the identifier circle in clockwise direction. The predecessor is counter-clockwise.

A Bruijn network is at its maximum performance when it is fully completed. Whenever the Bruijn graph is not complete, the search for a specific Profile Cluster can suppose a high number of hops. This is due to the fact that the Superpeers involved in the search path do not exist yet. At startup, the Bruijn network is created as peers with different Profile Vectors V enter the system. Thus, the Bruijn network is incomplete during this time. In order to improve this, a solution based on *virtualization of superpeers* is proposed. The objective is to maintain entire connection in the graph even when some nodes are missing. The virtualization of superpeers means that, apart from its own key, a physical superpeer (a Bruijn node) can have a continuous range of Profile Vectors assigned, and these identify non-existent superpeers in the net. Each physical superpeer, identified by the Hilbert transform P, stores its own Profile and all the k Profiles related to the virtual nodes between P and its physical predecessor node ($P - k$), following the Hilbert order. Thus, as the system grows, fewer virtual keys are assigned to physical superpeers. As an example, imagine that we have an incomplete Bruijn(2,4). If the graph were completed, it would be formed by 16 physical superpeers (the subindex indicates its order): $P_0 \ldots P_{15}$. Let us suppose that the graph is incomplete with for example, four physical nodes (P_4, P_7, P_{11}, P_{15}). This means that the rest of superpeers are virtual (not created yet) and the Profiles should be assigned to the physical ones. Then, physical node P_4 should contain the virtual nodes (P_0,P_3), P_7 (P_5, P_6), P_{11} (P_8,P_{10}) and P_{15} (P_{12},P_{14}).

On the other hand, when a node disappears from the network, the successor becomes responsible for its keys. For more details on how keys are assigned to nodes and how can the value for a given key be discovered, see [Cast 10, Cast 11a].

Building the Profile Cluster

A Profile Cluster contains peers that share a similar Profile Vector, V. The node in charge of its associated Profile cluster, responsible for managing the connection of their forming peers is its associated superpeer SP. The superpeer holds a FIFO queue with the information (physical addresses) of 5 peers of the Profile Cluster. When a new node P with a Profile Vector V arrives at the Profile Cluster, it contacts with the superpeer SP and then, the following steps are performed:

1. The new node P requests the FIFO peer list of physical addresses to SP.
2. SP delivers the FIFO list to P. SP updates its FIFO list. The address of the oldest peer in the list is removed and the new physical address of P is inserted.
3. P attempts to connect to σ peers of the Profile Cluster administered by SP. The σ value is ranged from $Kmin$ to $Kmax$ thresholds. Section 6.4 gives more details about the most suitable values for these threshold.
4. P waits until it receives the acceptance message from σ peers. Neighbors with $Kmax$ links will not accept connection.
5. P join the netwok. It can start executing the privacy protocol for submitting queries to the WSE.

Maintenance of the P2P connectivity

The maintenance of the P2P networks consists in detecting broken links. For this purpose, the system requires that users exchange messages every T units of time to know if their neighbours are still "alive".

Elapsed a period T, if a peer has not yet received the message notification "alive" from the expected neighbour, it executes the *departure mechanism* described below in the same Section. We must differentiate between departure of peers in the Bruijn and the Profile Cluster layers.

The value of T must be short enough to act quickly in case of peer disconnections. At the same time, T should be long enough not to saturate the network with the maintenance messages. For more details in obtaining the best value for T, see [Cast 11a].

When a peer leaves the system (voluntarily or involuntarily), it is necessary to restructure the broken links to maintain its connectivity and functionality. Otherwise, the performance of the network is critically reduced, increasing the number of steps required to locate a node, the number of links, and probably, producing isolated groups of peers. The disconnection of a superpeer and a simple peer affect the system in different manners. Next, the departure policy from each level is described.

When a superpeer SP leaves the network, the following procedure is performed:

1. As a result of the maintenance policy, after T units of time, a neighbouring superpeer detects a broken link.

2. As explained in Section 6.3.3, the successor of the superpeer SP (lets say SP_b) in the Bruijn network will be in charge of the Profiles of the leaving node.

3. In order to have the network fully connected, SP_b will create links with the neighbours of the leaving Superpeer.

4. Finally, SP_b sends a notify message to her neighbours informing about the changes.

If a peer of the Profile Cluster level leaves the system, the procedure is slightly different. The Profile Cluster peers are connected to the system during periods that we call "sessions". Every time that a peer ends a session, the σ neighbours that detect the broken link have to connect to another node, as long as the σ degree remains below the $Kmin$ threshold. The procedure for finding neighbours works as follows:

1. Selecting peers from the *FIFO* queue initially provided by the *superpeer SP*. If the list is empty, the second option is used.

2. Applying an uninformed searching algorithm (e.g. *Random Walk* [Lua 05, Sarm 07]) to select a random peer within the Profile Cluster.

6.3.4 Privacy-preserving protocol for submitting a query

This section explains the protocol that users execute inside the Profile Cluster in order to privately submit their queries. As previously mentioned, the protocol is based on the approach presented in [Viej 10]. However, some modifications are necessary in order to adapt it to a more dynamic scenario. More specifically, the modifications are mainly applied to the protection against selfish users, and the criteria to accept a query from a neighbour. The resulting protocol is divided into three different phases: the initializations, the transfer of a query, and the return of the results.

Initializations

We assume that the peers are already organized in their Profile Clusters after executing the steps described in Section 6.3.3. However, during this process, every time that P creates a link with a new neighbour Q, the next steps are executed:

1. P generates a random value r_p and keeps it secret.

2. P employs a cryptographic hash function $H()$ and obtains $\beta + 1$ different values as follows:

$$\begin{aligned} H(r_p) &= h^p_\beta \\ H(h^p_\beta) &= h^p_{\beta-1} \\ H(h^p_{\beta-1}) &= h^p_{\beta-2} \\ &\vdots \\ H(h^p_1) &= h^p_0 \end{aligned}$$

3. Similarly, Q executes steps 1 and 2 with a random value r_q and the same cryptographic hash function $H()$. After this, Q obtains β different values $\left\{h^q_\beta, h^q_{\beta-1}, \ldots, h^q_0\right\}$.

4. P sends h^p_0 to Q.

5. Q sends h_0^q to P.

P repeats this process for each one of her σ neighbours: $Q_1, Q_2, \ldots, Q_\sigma$. As a result of that, she obtains $h_0^{q_1}, h_0^{q_2}, \ldots, h_0^{q_\sigma}$.

In addition to this, P creates an array of records called *pending queries*. Each record has three fields: the text of the query, the address of the peer that sent the query to P, and the address of the peer to which P forwarded the query.

Transferring a query

When a peer P wants to submit a query, the following steps are executed:

1. P uses a random function to select the neighbour to which she will forward the query. The outcome of the function is one element at random $Q_f \in Q_1, Q_2, ..., Q_\sigma$. Then, P forwards the query and h_{i+1}^p to Q_f, where i is the number of queries that P and Q_f have already exchanged.

2. In order to decide whether Q_f should accept the query, the following steps are executed:

   (a) Q_f calculates $H(h_{i+1}^p)$ and checks that the result equals h_i^p.

   - If $(H(h_{i+1}^p) \overset{?}{=} h_i^p)$:

     i. Q_f calculates $s = H(h_{i+1}^p || h_i^{q_f})$, where $||$ denotes the concatenation operator.

     ii. Q_f employs s as a seed for a pseudorandom generator $a = Rand(s)$, where $a \in [0, \ldots, 1]$ is the probability of accepting the query.

     – If $a < \lambda$, Q_f rejects the query and sends (NO, $h_{i+1}^{q_f}$) to P.

     – If $a \geq \lambda$, Q_f accepts the query and sends (OK, $h_{i+1}^{q_f}$) to P.

     Where λ is a parameter of the system that represents the query acceptance ratio.

   - Otherwise, Q_f detects a misbehaviour and ends the connection with P. The protocol finalizes at this point.

   (b) If P receives NO from Q_f, she must check if the rejection is justified or ir Q_f is behaving selfishly. For this purpose, P calculates $H(h_{i+1}^{q_f})$ and checks that the result equals $h_i^{q_f}$.

- If $(H(h_{i+1}^{n_f}) \overset{?}{=} h_i^{n_f})$,
  P calculates $s = H(h_{i+1}^{p} || h_{i+1}^{q_f})$ and $a = Rand(s)$ in the same way that Q_f did it.
  - If $a \geq \lambda$, P realizes that Q_f should have accepted her query and that Q_f is behaving in a selfish manner. Then, she ends the connection with Q_f and the protocol finalizes at this point.
  - If $a < \lambda$, P realizes that the rejection was justified. Then, P may try to send her query to one of her other neighbours.
- Otherwise, P detects a misbehaviour and ends the connection with Q_f. The protocol finalizes at this point.

(c) If P receives OK from Q_f, she saves in the *pending queries* array the text of the query, her own address to indicate that the query was generated by her, and the address of Q_f.

3. Assuming that P has found a neighbour Q_g that accepts her query, Q_g must determine if she submits the query of forwards it to one of her neighbours $Q'_1, Q'_2, ..., Q'_\sigma$. Note that since Q_g is not the user who generated the query, she is also a potential submitter. The decision of submitting of forwarding the query is again based on a random function. In this case, the outcome of the function is one element at random inside the set $Q_g, Q'_1, Q'_2, ..., Q'_\sigma$. If the function returns Q_g, the query is submitted by Q_g and this phase of the protocol ends.

   Otherwise, the query is forwarded to the neighbour $Q'_g \in Q'_1, Q'_2, ..., Q'_\sigma$ that the function returns. Then, Q'_g must again execute Step 2 to decide if she accepts the query.

4. Previous steps are repeated until eventually any peer submits the query to the WSE.

Note that this protocol allows each pair of peers to exchange a maximum of β queries. For the $(\beta+1)$-th query, a peer P and her neighbour Q must execute again the initialization steps and obtain new values $\left\{h_\beta^p, h_{\beta-1}^p, \ldots, h_0^p\right\}$ and $\left\{h_\beta^q, h_{\beta-1}^q, \ldots, h_0^q\right\}$, respectively.

Return of the results

After the query is submitted by a peer inside the Profile Cluster, the results must be sent back to the user who generated the query. However, the user who submitted the query does not know the peer who generated it. For this reason, she employs the *pending queries* array to know the address of the peer that sends that query to her. She sends the results to that peer, who sends it to the user that forwarded it to her, and that user to the previous one, etc. Everytime that a peer forwards the search results from a query, she removes the corresponding record in the *pending queries* array.

The result of this process is that, in order to reach the origin, the search results follow the inverse path that the query employed. However, let us consider that one of the nodes in this path fails. This would prevent the peer who generated the query from obtaining the search results.

In order to avoid this problem, a resilience against failures protocol is executed. This protocol is a simple solution that profits the maintenance protocol of the P2P network described in Section 6.3.3. In this protocol, the users receive "alive" messages from their neighbours every period T. If after a period T, a peer does not receives a message from a neighbour, she concludes that the node has left the system.

In this case, she must check her *pending queries* array to know if she had a pending query $pend_q$ forwarded to the node that failed. If this is the case, the peer P has two options:

1. Executing again the second phase of the protocol (see Section 6.3.4), and forwarding $pend_q$ to another of her remaining neighbours. This also implies updating the *pending queries* array.

2. Consulting the *pending queries* array to know who was the neighbour Q_i that forwarded $pend_q$ to her. Then, she sends a message to Q_i indicating that the path for $pend_q$ has been truncated.

In the second case, the receiving node has again the two previous options. At the end, the query would be submitted following a different path or the originator would receive a message of path truncated.

6.3.5 Protocol behavior in front of the considered adversaries

This section analyzes the behavior of the proposed protocol in front of the adversaries defined in Section 6.2.

Against the WSE

As explained before, the peers are organized in Profile Clusters where they are connected to other peers that share the same interests. Inside a Profile Cluster, they exchange their queries executing the proposed privacy protocol. Regarding the WSE, this has two consequences:

1. Since a peer submits queries on behalf of other users, the WSE cannot distinguish which query belongs to each user. Therefore, the privacy of the peer is protected. The privacy level that peers obtain is comparable to the use of k-anonymity [Swee 02] (in our case, σ-anonymity).

2. Since a peer submits queries that belong to users with similar interests, the profile that the WSE owns is still reliable. This allows the peers to obtain *personalized* search results, i.e., results that match their interests.

Against a dishonest user

Dishonest users try to learn the queries that belong to other users. The only information that they have for this purpose is the query associated to the peer that forwarded it. However, a dishonest user who receives a query cannot discern if the query was generated by that peer or if that peer was forwarding it on behalf of another user. The dishonest user knows neither the topology of the Profile Cluster nor the number of hops from the legitimate owner of the query to the final node. Additionally, the system is dynamic, with users constantly joining and leaving the system. All this prevents her from finding the real source of a query.

Against a selfish user

As explained in Section 6.2.1, the system must penalize users that behave in a selfish way. Since the connections between users are dynamic, the protection against selfish users is limited and it can only be applied to a current connection. This means that the proposed method can protect a peer P that is connected to a selfish neighbour Q. However, if Q leaves and re-joins the system, it can no longer be identified as a previously selfish user. The new peers that are connected to Q will have to re-identify it as a selfish user. The method proposed for this purpose is described in Step 2 from Section 6.3.4.

6.4 Performance evaluation

In order to analyze the performance of the proposed architecture, the system has been implemented and simulated. This section explains the results obtained in these simulations.

6.4.1 Methodology and test environment

First of all, we describe the points of the system that are going to be evaluated in this section:

- *Time to join the system* In a P2P protocol where a group of users are connected between them, the time required to join the system is an important point. For a new peer, this period of time starts when she contacts the bootstrap, and ends when she is in the Profile Cluster that best matches her profile, connected to a minimum number of neighbours.

- *Degree distribution* This point analyzes the degree of the peers in the profile clusters (i.e., the number of neighbours that each node has). The peers in the network execute the privacy preserving protocol described in Section 6.3.4. The higher the degree of a peer, the more privacy she obtains, but the higher overload she gets. Additionally, having peers with very different degrees prevent dishonest users from knowing the topology of the network. On the other hand, it is not desirable to leave any peer isolated.

Prior to the simulations, there are three parameters that must be configured in the system:

- *The size of the Bruijn network*. This parameter refers to the number of Bruijn superpeers in the network. It also represents the maximum number of different profiles that the architecture supports. As it is defined in [Kaas 03], in a d-dimensional Bruijn graph, there are 2^d nodes. This means that the number of nodes is always a power of 2. For the tests, we have evaluated different dimensions of Bruijn, with 4, 8, 16, and 32 nodes ($d = 2, 3, 4, 5$).

- *Kmin*. This parameter is the minimum number of neighbours that a peer must have in any moment. If one or more of her neighbours disconnect, and the peer detects that she has less than *Kmin* connexions, she must contact her superpeer in order to create new links. The peer will repeat this process until her degree is at least *Kmin*. Note that the privacy preserving protocol described in Section 6.3.4 requires that $Kmin \geq 2$. Otherwise, a peer P who has a single neighbour Q would lose her privacy. In such scenario, P would be forced to send all her queries to Q, and Q would know that all the queries received from P were generated by her. Therefore, the higher the degree of a peer, the more privacy she obtains. On the other hand, forcing the peers to have a high number of minimum neighbours may increase the load of the network. For the tests, we have evaluated different values of $Kmin = 2, 3, 10$.

- *Kmax*. This parameter is the maximum number of neighbours that a peer can have. If a peer already has *Kmax* neighbours, she cannot accept new connections. For most of the tests, we have employed a $Kmax = \infty$ (i.e., we have not limited the maximum degree of the peers). Then, we have performed a test with $Kmin = 2, Kmax = 4$, in order to evaluate how limiting *Kmax* affects the system.

In order to perform the simulations, a tool called PlanetSim [Ahul 08] has been employed. PlanetSim is a simulation framework for overlay networks and services that allows developers to evaluate their own protocols. By using the API that PlanetSim provides, the P2P architecture defined in Section 6.3 has been implemented. Additionally, to simulate users' behaviours in the network, real data generated by AOL users have been used. These data have been extracted from the AOL dataset [M Ba 05]. As previously described, the AOL dataset contains the queries submitted by each AOL user during three months, and the time at which each query was submitted. Figure 6.3 shows an example of the volume of queries submitted to AOL per second, during a period of 3 days. The graph presents three peaks corresponding to a similar number of queries each day at the same time. It can be observed that the behaviour of AOL users during one day recurs the other days.

For our simulations, a one-week period file from the AOL dataset has been extracted and tested, as if the users had been using the P2P architecture. Therefore, the system is tested using real delays obtained from real users. Note that since the

AOL dataset does not contain specific information about user profiles, we have assigned random profiles to each AOL user.

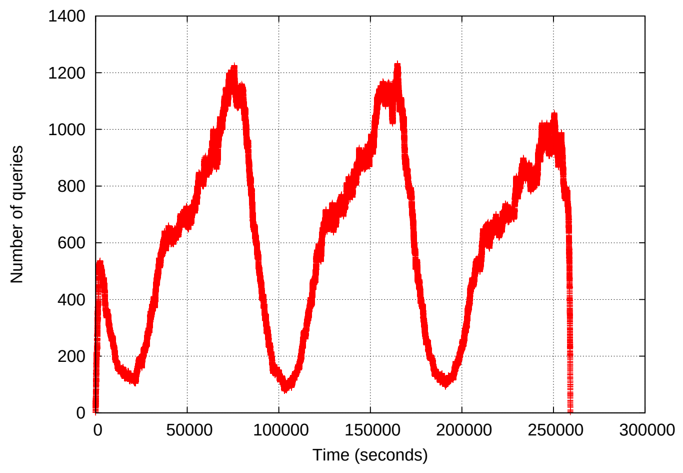


Figure 6.3: Distribution of submitted queries to AOL during 3 days

6.4.2 Time to join the system

As explained above, the first test evaluates the time required to join the system. We simulated one week of real data, using an application implemented with PlanetSim. The results obtained reflect the number of tics that each user needs in order to find her place in the network, and create the links with her neighbours.

From the results obtained, we initially detected the presence of outliers, i.e., observations that appear to deviate markedly from other members of the sample in which it occurs [Grub 69]. Outliers can significantly affect the estimation of statistical parameters such as the mean or the standard deviation in a dataset. These outliers must be eliminated in order to obtain reliable estimators. There are several methods for detecting outliers [Barn 94], but the simplest ones are those that employ (i) the interquartile range (difference between quartile 3 and quartile 1) or (ii) the variability of the data from its mean. In our case, it was not possible to employ the first criteria, since in the result set quartile 1 and quartile 3 were frequently equal. On the other hand, the results do not follow a normal distribution: data are not symmetrically distributed, and they have a strong bias. As a result, we decided to apply the *Chebyshev's inequality* to detect and remove outliers, since this method does not assume that the data follow any particular distribution.

Chebyshev's inequality guarantees that no more than $100/k^2$ % of the distribution's values can be more than k standard deviations away from the mean. In our case, we fix $k = 8$ and we consider any observation that has more than 5 standard deviations of the mean to be an outlier. According to *Chebyshev's inequality*, this process affects 1.56% of the observations at most.

The method has been executed iteratively until all observations have less than 5 standard deviations of the mean. Accordingly, we have obtained the results presented in Table 6.3, which contains the average number of tics that the peers employ to join the system and their standard deviation. The results are presented for different sizes of Bruijn and different $Kmin$, while $Kmax = \infty$ in all the cases.

Table 6.3: Average tics and standard deviation for joining the system for $Kmax = \infty$

	$Kmin = 2$		$Kmin = 3$		$Kmin = 10$	
	mean	dev.	mean	dev.	mean	dev.
4 Bruijn nodes	11.324	1.501	12.826	1.357	20.260	1.326
8 Bruijn nodes	11.957	1.739	13.463	1.632	21.260	1.692
16 Bruijn nodes	12.553	2.060	14.058	2.021	22.228	2.299
32 Bruijn nodes	12.326	1.920	13.827	1.880	21.978	2.188

From the results presented in Table 6.3, two statements can be inferred:

1. The system is scalable with respect to the number of Bruijn nodes. The time to join the system is maintained when the number of Bruijn nodes (and therefore, the number of different supported profiles) increases. Table 6.3 reflects that for the same value of $Kmin$, the average number of tics to join the system is maintained, regardless of the number of Bruijn nodes.

2. The time to join the system increases when $Kmin$ is augmented. This happens because in order to join the system, a peer is forced to initially find $Kmin$ neighbours. Then, the higher the $Kmin$, the longer the peer spends looking for neighbours. Applying a linear regression to the results of Table 6.3 reveals that each peer requires on average 1.15 tics more per extra neighbour.

Table 6.4: Average tics and standard deviation for joining the system for $Kmin = 2$, $Kmax = 4$

	$Kmin = 2$. $Kmax = 4$	
	mean	dev.
4 Bruijn nodes	20.073	23.581
8 Bruijn nodes	30.633	50.777
16 Bruijn nodes	53.230	108.180
32 Bruijn nodes	52.942	107.853

Time to join the system with a bounded *Kmax*

Results for previous experiments were calculated for a $Kmax = \infty$. However, it is also interesting to analyze what happens when *Kmax* is fixed to a particular value. Table 6.4 shows the average number of tics to join the system and their standard deviation, for $Kmin = 2$ and $Kmax = 4$.

A comparison between the results for $Kmin = 2$, $Kmax = \infty$ (Table 6.3) and the results for $Kmin = 2$, $Kmax = 4$ (Table 6.4), reveals that:

1. Limiting *Kmax* significantly increases the average number of tics to join the system. When the peers that are already in the system are not allowed to accept more than $Kmax = 4$ connections, it is more difficult for the new peers to find $Kmin = 2$ neighbours.

2. The standard deviation is higher for $Kmax = 4$ than for $Kmax = \infty$. This is due to some situations that happen in the network when *Kmax* is limited. For example, when several peers join or leave at the same time, there are many peers looking for new connections simultaneously. On the other hand, some Profile Cluster may have users with a small number of connections in a particular moment, so the peers are more susceptible to aacept new connections. As a consequence of these situations, the peers obtain delays to join the system which significantly deviates from the average value.

3. When *Kmax* is limited, the number of Bruijn nodes affects the average number of tics to join the system. Increasing the number of Bruijn nodes means to increase the number of Profile Clusters, so the peers are more dispersed in the network. With fewer peers per Profile Cluster, when these peers can only accept $Kmax = 4$ connexions, it is more difficult for new peers to find

Table 6.5: Average degree and standard deviation of the peers

	$Kmin = 2$		$Kmin = 3$		$Kmin = 10$	
	mean	dev.	mean	dev.	mean	dev.
4 Bruijn nodes	4.013	1.984	5.866	2.510	15.262	4.002
8 Bruijn nodes	3.943	1.859	5.748	2.370	14.940	3.610
16 Bruijn nodes	3.813	1.709	5.579	2.208	14.062	3.137
32 Bruijn nodes	3.811	1.708	5.560	2.207	14.161	3.124

$Kmin = 2$ neighbours. However, this is not a problem when $Kmax$ is not limited, since the peers, dispersed or not, can accept ∞ connexions.

6.4.3 Degree distribution

In order to evaluate the degree distribution of the network, the degree of each node at each moment has been calculated. Table 6.5 shows the average number of neighbours per node and the standard deviation. As in the previous test, the results are presented for different sizes of Bruijn and different $Kmin$, while $Kmax = \infty$ in all the cases.

Table 6.5 reflects that, as expected, the higher $Kmin$ is, the more average connexions each peer has. Regarding the number of Bruijn nodes, they slightly affect the average degree. Having more Bruijn nodes means that there are fewer peers inside each Profile Cluster and consequently, the peers have fewer neighbours.

Degree distribution with a bounded *Kmax*

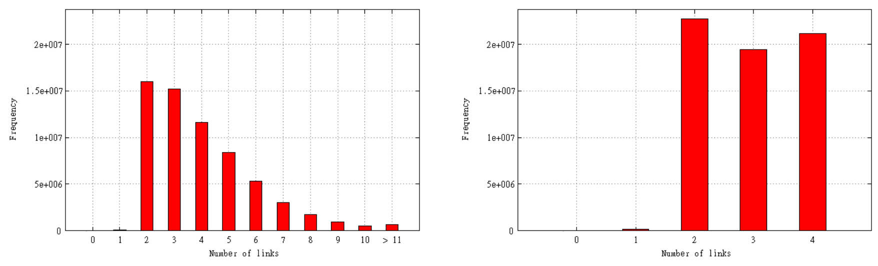
Results presented in Table 6.5 are calculated for $Kmax = \infty$. However, when analyzing the degree distribution, a comparison between the average number of neighbours when $Kmax$ is limited and when it is not is necessary. Table 6.6 shows the average number of links per user and the standard deviation, for $Kmin = 2$ and $Kmax = 4$.

From the results for $Kmin = 2$, $Kmax = \infty$ (Table 6.5) and the results for $Kmin = 2$, $Kmax = 4$ (Table 6.6), we can observe that:

1. Limiting $Kmax = 4$ reduces the average number of links that each peer has. With this restriction, peers are not allowed to have more than $Kmax = 4$ neighbours. Therefore, the mean does not have a bias towards the right, as it happens when $Kmax = \infty$.

Table 6.6: Average degree and standard deviation for $Kmin = 2$, $Kmax = 4$

	$Kmin = 2.$ $Kmax = 4$	
	mean	dev.
4 Bruijn nodes	2.964	0.836
8 Bruijn nodes	2.935	0.833
16 Bruijn nodes	2.874	0.835
32 Bruijn nodes	2.878	0.834



(a) Degree distribution for $Kmin = 2$, $Kmax = \infty$

(b) Degree distribution for $Kmin = 2$, $Kmax = 4$

Figure 6.4: Histograms for the degree distribution

2. Limiting $Kmax = 4$ reduces the standard deviation from the average number of links that each peer has. Fixing $Kmin = 2$, $Kmax = 4$ forces the peers to have 2, 3, or 4 neighbours. On the other hand, with $Kmin = 2$, $Kmax = \infty$ peers are allowed to have any number of neighbours in the range $[2, \ldots, \infty]$. Consequently, there are many peers with a degree that significantly differs from the average value (i.e., a higher standard deviation).

   Figure 2 shows a graphical representation of the degree distribution for $Kmin = 2$, $Kmax = \infty$ and for $Kmin = 2$, $Kmax = 4$. Figure 6.4(a) shows a degree distribution among several different values. The most frequent values correspond to 2 and 3 links per peer. Then, the higher the degree is, the lower frequency it has. On the other hand, Figure 6.4(b) shows a more dense degree distribution, with many peers that share the same number of links.

3. Regarding the privacy-preserving protocol from Section 6.3.4, $Kmax = 4$ offers a better load balance while $Kmax = \infty$ offers a higher level of privacy. As explained in Section 6.3.4, peers that execute the privacy-preserving protocol forward queries to their neighbours. When $Kmax = \infty$ peers with higher degree may be overloaded, while peers with lower degree may have long

periods of inactivity. When $Kmax = 4$, all the peers have a similar number of neighbours and hence, the load is balanced. However, the problem when $Kmax = 4$ is that a peer knows that each of her neighbours has 1, 2, or 3 other neighbours. As stated in Section 6.3.5, dishonest users cannot find the source of a query because they ignore the topology. Giving information about the topology may threaten the privacy of honest users.

6.5 Privacy Evaluation

While Section 6.4 focuses on the performance analysis of the Bruijn architecture, this section presents a privacy evaluation of the protocol described on Section 6.3.4. In this case, the objective of the simulations is to show that:

- The protocol can successfully preserve honest users' privacy in front of the WSE.
- The protocol penalizes selfish users by exposing them to the WSE.
- Employing the proposed special-purpose network with dynamic groups outperforms the results obtained in [Viej 10], which executes a similar privacy-preserving protocol but deployed in a social network.

First of all, we present the conditions that determine if a user is exposed in front of the WSE. For this purpose, we adapt the definition of an exposed user presented in [Viej 10] to our scenario:

Definition 6.1. *(Exposed user) Let:*

- $G = \{U_1, U_2, \ldots, U_n\}$ *be the group of users of a WSE.*
- q *be a certain query.*
- z_i *be the number of times that a certain user* U *has submitted the same query* q_i.
- $L_U = \{(q_1, z_1), (q_2, z_2), \ldots, (q_r, z_r)\}$ *be the list of queries submitted by* U.

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

Therefore, there are two requirements in order to guarantee that a certain user U *is not exposed:*

- *Requirement 1. Given a certain query q, the WSE can group all the users U_i who satisfy $(q, z_i) \in L_{U_i}$. Then, the WSE orders that group depending on the number of times z_i that each user has submitted q. This is $Y = \{U_1, U_2, \ldots, U_k\}$, where $(q, z_i) \in L_{U_i}$, $(q, z_{i-1}) \in L_{U_{i-1}}$, and $z_i > z_{i-1}$ for $\forall i \in Y$. Let us consider that U_j was the user who initially generated q, i.e., $(q, z_j) \in L_{U_j}$. The maximal uncertainty for the WSE happens when all the users $U_i \in Y$ have submitted q the same number of times. This is $z_i = z_j$ when $i \neq j$, for $\forall i, j \in Y$. This is a very strict requirement, and it can be difficult to achieve. Therefore, we relax it and we only require z_j to be a value in the range $z_1 \leq z_j \leq z_k$. This requirement prevents z_j from being the only value out of range. If z_j is out of range, the WSE can easily identify U_j and hence, she is exposed.*

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

A user U that fulfills the requirements above is concealed into a group of k users and achieves a level of privacy comparable to k-anonymity. This implies that the probability of correctly identifying U is at most $1/k$. Note that, theoretically, any user who submits q to the WSE is considered in the group of k users. In our scenario, these are the peers inside a profile cluster that have ever submitted a query generated by U.

Next, we describe the performed simulation tests:

- The network inside a profile cluster was modeled using a random distribution. Initially, each user is connected to a number of neighbours ranging between 2 and 10, but the number of connections may vary during the simulation. This is one of the differences with respect to [Viej 10], which employed a power-law distribution (followed by typical social networks) with static connections between users.

- Similarly to [Viej 10], and in order to compare the obtained results in both systems, we have performed simulations considering networks of 50, 100, 200 and 400 users.

- Another difference with respect to [Viej 10] is that our system considers users that join and leave the system. In order to accurately simulate this behaviour, we have employed real data from the AOL dataset. First, we have extracted four data subsets where only 50, 100, 200 and 400 users submit their queries. Then, we have divided the data in sessions. A session is defined as a group of queries made by a single user for a single navigation purpose [Huan 04]. The most commonly used session identification method is called timeout [He 00], in which a user session is usually defined as a sequence of queries from the same user such that two consecutive queries are separated by an interval less than a predefined threshold. In our case, the sessions are divided using a threshold set arbitrarily to 3600 seconds. After that, we have calculated the percentage of time in which each user has an open session with respect to the duration of its data subset. These percentages have been employed as the probability of being online in a particular moment. For each step of the simulation, each node has been marked as online or offline based on this probability.

- A user U that has a 100% probability of being online, generates 1656 queries (in December 2012, each European user submitted, on average, 138 queries to a search engine [Stat 13], i.e., 1656 queries per year approximately). Users with lower online probability proportionally decrease their number of generated queries. All the queries generated by the same user have the same content. This is the worst case scenario, since it is easier to identify a certain user who continually submits the same query.

- Tests have been performed with different values for the acceptance ratio parameter. This parameter, employed at Step 2(a)ii in the protocol of Section 6.3.4, determines the probability of accepting or rejecting the query received from a neighbour.

- As in [Viej 10], we have run two kinds of simulations with two different kinds of users: honest users and selfish users. Dishonest users have not been considered because they follow the protocol in the same way that honest users do. The only difference is that dishonest users try to profile other users during the process.

6.5.1 Tests without selfish users.

The first kind of tests is run in a scenario where all the users are honest users. The objective of the tests is to analyze the average number of hops that a certain query does before being submitted to the WSE, and the percentage of exposed users.

For this purpose, we have simulated four different networks consisting of, respectively, 50, 100, 200, and 400 users. For each network, 1000 simulations have been run and the average of these results has been computed. Results show that:

- Figure 6.5 shows the average number of hops for each of the networks, employing different acceptance ratios (probabilities 0.3, 0.5, 0.7, and 0.9 for a user to accept the query from a neighbour). The results shows that the number of hops increases with the number of users and with the acceptance ratio. This means that the bigger the network is and the more likely the neighbours are to accept a query, the further the query arrives.

  However, in comparison with [Viej 10], the average number of hops that the authors obtained (2.4 hops for a social network of 400 users) has significantly increased. This is caused by the more dynamic nature of our network as opposed to the static nature of a social network. In our scenario, users with high online probability tend to be connected to a higher number of neighbours, specially to neighbours who also have a high online probability. As Step 3 of the protocol presented in Section 6.3.4 establishes, the probability of submitting the query to the WSE is $1/(\sigma+1)$. Therefore, users with many neighbours (i.e., higher σ) are more likely to forward the query, and hence, this increases the average number of hops of the overall network.

  A higher number of hops has its advantages and disadvantages. On one hand, it increases the response time, since the user needs to wait longer for the results. On the other hand, it provides a higher anonymity, since it is harder for an external observer to track back the query to its initial source. One simple solution for scenarios where response time is critical would be to increase the probability of submitting the query to the WSE to $\rho/(\sigma+1)$, where ρ is a configurable parameter of the system.

- Another result obtained from the tests without selfish users is a 0% of exposed users in all the networks. This significantly outperforms the 58% of

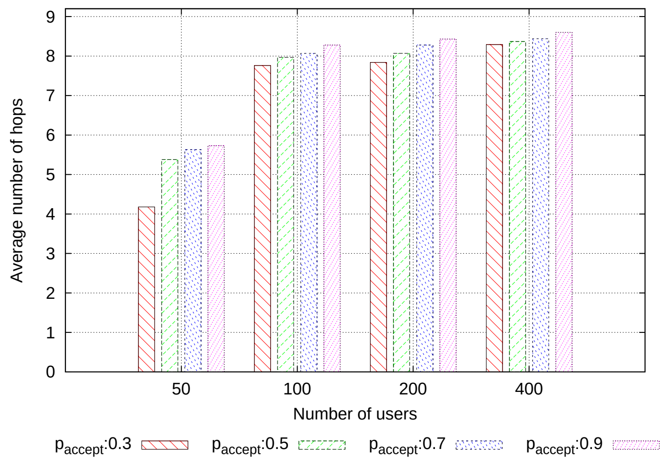


Figure 6.5: Average number of hops for different acceptance ratio

exposed users in a social network of 50 users, or the 43.5% of exposed users in a social network of 400 users obtained in [Viej 10]. There are two reasons that explain these results. The main one is that the dynamic nature of the network induces users to have different neighbours in different moments of the simulations, which causes a more even and sparse distribution of any query among the users of the network. Another reason for the lack of exposed users is that the privacy-preserving protocol, by default, prevents any user from submitting her own query (unless she finds no neighbour that accepts it or if the query follows a cycle). This significantly reduces the number of times that a user submits her own query with respect to the protocol of [Viej 10], improving the compliance with the first requirement in the definition of an exposed user.

6.5.2 Tests with selfish users.

This kind of tests are meant to analyze the behaviour of the system in presence of selfish users. More specifically, the objective is to show how the percentage of exposed users changes when selfish users participate in the protocol. For the sake of simplicity, the simulations have only been performed using a network of 400 users and an acceptance ratio $p_{accept} = 0.7$. The same four different scenarios considered in [Viej 10] and in our tests are:

- **Scenario 1**: 90% honest users, 10% selfish users.

- **Scenario 2**: 80% honest users, 20% selfish users.
- **Scenario 3**: 60% honest users, 40% selfish users.
- **Scenario 4**: 20% honest users, 80% selfish users.

Table 6.7 reflects the behaviour of the protocol in the four scenarios. Results show that, for all the configurations, no honest user has been exposed. Regarding selfish users, between 65% and 70% of them have been exposed. Additionally, the third and the fourth columns show that, for a selfish user, the exposition depends on the amount of time that they have remained online. It can be observed that selfish users who frequently employ the system are exposed, while selfish users who rarely submit queries remain unexposed. Consequently, these results show that the system correctly penalizes selfish users who regularly employ the system to submit their queries.

In comparison with [Viej 10], results show that their scheme obtains a higher number of exposed honest users, specially for users that have fewer connections. For example, in Scenario 1, honest users with 7, 8, 9, and 10 neighbours are never exposed, while honest users with 1 or 2 neighbours have a percentage of exposed users higher than 90%. For Scenarios 2, 3, and 4, the percentages of exposure increase for all the honest users.

Table 6.7: Percentages of exposed users and average online time for selfish users.

	Exposed honest users	Exposed selfish users	Average online time for exposed selfish users	Average online time for non-exposed selfish users
Scenario 1	0%	70%	49.31%	1.25%
Scenario 2	0%	67.25%	47.40%	0.66%
Scenario 3	0%	65.63%	42.84%	0.44%
Scenario 4	0%	64.94%	39.51%	0.24%

6.6 Conclusions

In this chapter we have presented a system that protects the privacy of WSE users, while maintaining the quality of service that the WSE can offer. For this purpose, we have designed a P2P architecture that supports a privacy-preserving protocol for WSE users. Basically, the system classifies users into groups according to their interests, and the privacy-preserving protocol allows them to obtain a *group profile*. As a result, the WSE can employ the *group profile* to improve the user's experience, but it ignores detailed person-specific data.

The privacy-preserving protocol protects users' privacy in front of the WSE and of dishonest users, and it also penalizes users that behave selfishly. The P2P architecture has been implemented and simulated. Results regarding the time to join the system and degree distribution have been presented for several network configurations. The simulations have been performed using real data belonging to a large number of users, which shows that the system works even with a high load of users. Additionally, results show that the network manages the joins and leaves of users and hence, (i) users can join the network and find $Kmin$ neighbours with a reasonable delay, and (ii) when a user leaves the system, the connectivity of her neighbours is maintained, and no node remains isolated. Regarding the degree distribution, results show that if $Kmax$ is not bounded, users have different number of neighbours. As a consequence, the topology of the network is unpredictable, protecting users' privacy in front of dishonest internal users.

As for future work, it might be interesting to simulate how the protocol behaves in terms of performance for different sizes of $Kmin$ and $Kmax$. Additionally, it might be interesting to observe how the *group profiles* evolve and how they differ from the profiles that the users would have obtained if they had not executed the protocol.

Chapter 7
Conclusions

This chapter summarizes the contributions, the related publications and describes possible future research lines.

Contents

7.1 Contributions

In this thesis, we have focused on providing client-side solutions for private web search. On one hand, we have performed a survey of the current literature, classifying the proposals, describing their advantages and disadvantages, and comparing them according to some properties.

On the other hand, we have contributed with three proposals that protect users' privacy in front of WSEs. These three proposals have been analyzed in terms of privacy achieved and performance. More specifically, the contributions regarding these three proposals are:

1. First of all, we have presented a protocol that reduces the query delay regarding similar proposals in the literature. Additionally, another novelty of the protocol is that it incentivizes every user to follow the protocol in order to obtain privacy. The scheme has been tested in an open environment and results show that it achieves the lowest query delay which has been reported in multi-party private web search protocols.

2. Despite the high performance of the previous protocol, it is not prepared to prevent malicious internal attacks. For this reason, our second proposed

protocol is adapted to work in a more hostile scenario, addressing the problem of dishonest users. The scheme has been implemented, compared with similar proposals, and evaluated using real queries in a controlled and real environment. Results show that it outperforms proposals with a similar privacy level, and that it preserves users' profiles at general categories, while protecting its specific contents.

3. Although this second protocol is resilient against dishonest internal users, it does not protect users from a dishonest central entity responsible for grouping users (the central node). For this reason, the third proposed protocol is a peer-to-peer solution that does not require the use of a central node in order to create the groups. Basically, this proposal classifies users into groups according to their interests. Then, a privacy-preserving protocol is executed and users obtain a *group profile*. This group profile allows WSEs to personalize the results of each user, but ignoring detailed person-specific data.

   The P2P protocol has been implemented and simulated for several network configurations using real data. Results show that the delays to join or leave the network are affordable, and that the system behaves properly even with a high load of users. Additionally, results show that the protocol protects users' privacy in front of the WSE and of dishonest users, and it also penalizes users that behave selfishly.

7.2 Publications

The publications supporting the content of this thesis are stated below:

- Cristina Romero-Tris; Jordi Castellà-Roca; Alexandre Viejo "Distributed System for Private Web Search with Untrusted Partners" In *International Journal of Computer and Telecommunications Networking (Computer Networks)* Vol. 67, pp. 26–42. 2014.

- Cristina Romero-Tris; Alexandre Viejo; Jordi Castellà-Roca "Client-Side Privacy Enhancing Technologies for Web Search" In *International Journal for the Computer and Telecommunications Industry (Computer Communications)* 2013. Under review.

- Cristina Romero-Tris; Damià Castellà Martínez; Alexandre Viejo; Jordi Castellà Roca; Francesc Solsona Tehas; Josep Maria Mateo-Sanz "Design of a P2P network that protects users' privacy in front of Web Search Engines" Special Issue on Recent Advances in Security and Privacy in Distributed Communications In *International Journal of Computer and Electrical Engineering* 2013. Under review.

- Cristina Romero-Tris; Alexandre Viejo; Jordi Castellà Roca "Multi-party methods for privacy-preserving web search: survey and contributions" In *Advanced research in data privacy*. Springer, in press, 2014.

- Cristina Romero-Tris; Alexandre Viejo; Jordi Castellà Roca; Youssef Benkaryouh "Sistema P2P de protección de la privacidad en motores de búsqueda basado en perfiles de usuario" In *XIII Reunión Espanola de Criptografía y Seguridad de la Información (RECSI'14). (XIII Spanish Meeting on Cryptography and Information Security)*. 2014. To appear.

- Cristina Romero-Tris; Jordi Castellà-Roca; Alexandre Viejo "Privacidad en Motores de Búsqueda con un Protocolo Multi-usuario con Atacantes Internos" In *XII Reunión Espanola de Criptografía y Seguridad de la Información (RECSI'12). (XII Spanish Meeting on Cryptography and Information Security)*. 2012.

- Damià Castellà Martínez; Cristina Romero-Tris; Alexandre Viejo; Jordi Castellà Roca; Francesc Solsona Tehas; Francesc Giné de Sola "Diseño de una red P2P optimizada para la privatización de consultas en WSEs" In *XII Reunión Espanola de Criptografía y Seguridad de la Información (RECSI'12). (XII Spanish Meeting on Cryptography and Information Security)*. 2012.

- Cristina Romero-Tris; Jordi Castellà-Roca; Alexandre Viejo "Multi-party private web search with untrusted partners". In *Security and Privacy in Communication Networks, 7th International Conference on Security and Privacy in Communication Networks (SecureComm)* Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Volume 96, pp. 261-280. Springer Berlin Heidelberg. 2012. Core A.

- Cristina Romero-Tris; Alexandre Viejo; Jordi Castellà-Roca "Improving Query Delay in Private Web Search" In *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011. IEEE Computer Society, Washington, DC, USA, 200-206. DOI=10.1109/3PGCIC.2011.61 http://dx.doi.org/10.1109/3PGCIC.2011.61* . 2011.

7.3 Future work

In this section, we outline some of the possible new projects or open problems in order to make contributions in the area.

The work presented in this thesis is an ongoing effort where each proposal is improved upon over time. The proposal presented in Chapter 5 improves the proposal of Chapter 4 by making it resilient against internal attackers. The proposal presented in Chapter 6 improves the proposal of Chapter 5 by removing the central node, which may be a bottleneck and a dangerous adversary. However, there is still room for improvement and some points of this latter proposal must be addressed:

- It is required to study an efficient way of grouping users according to their interests. This is necessary in order to find their best position in the network.

- Another objective regarding this proposal is the simulation of the protocol in terms of performance for different sizes of $Kmin$ and $Kmax$.

- Additionally, it might be interesting to observe how the *group profiles* evolve and how they differ from the profiles that users would have obtained if they had not executed the protocol.

Regarding this last point, another line of future work is the study of efficient metrics to evaluate the level of privacy achieved when using a privacy-preserving web search scheme.

Bibliography

[Abe 99] M. Abe. “Mix-networks on permutation networks”. In: L. N. in Computer Science, Ed., *Advances in Cryptology – Asiacrypt'99*, pp. 258–273, 1999.

[Ahul 08] J. P. Ahulló and P. G. López. “PlanetSim: an extensible framework for overlay network and services simulations”. In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, pp. 45:1–45:1, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 2008.

[Al R 12] R. Al-Rfou', W. Jannen, and N. Patwardhan. “TrackMeNot-so-good-after-all”. Nov. 2012. arXiv :211.0320.

[Anto 10] A. I. Antón, J. B. Earp, and J. D. Young. “How Internet Users' Privacy Concerns Have Evolved since 2002”. *IEEE Security and Privacy*, Vol. 8, No. 1, pp. 21–27, Jan. 2010.

[Aram 12] A. Arampatzis, P. Efraimidis, and G. Drosatos. “A query scrambler for search privacy on the internet”. *Information Retrieval*, pp. 1–23, 2012.

[Bals 12] E. Balsa, C. Troncoso, and C. Diaz. “OB-PWS: Obfuscation-Based Private Web Search”. In: *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pp. 491–505, IEEE Computer Society, Washington, DC, USA, 2012.

[Barn 94] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, 3rd Ed., 1994.

[Beck 12] M. Beck and M. Marhfer. “Do-Not-Track Techniques for Browsers and Their Implications for Consumers”. In: J. Camenisch, B. Crispo, S. Fischer-Habner, R. Leenes, and G. Russello, Eds., *Privacy and Identity Management for Life*, pp. 187–196, Springer Berlin Heidelberg, 2012.

[Bert 01] O. Berthold, H. Federrath, and S. Köpsell. "Web MIXes: a system for anonymous and unobservable Internet access". In: *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*, pp. 115–129, Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[Bohm 04] R. Bohme, G. Danezis, C. Diaz, S. Kopsell, and A. Pfitzmann. "Mix Cascades vs. Peer-to-Peer: Is One Concept Superior?". In: *In Privacy Enhancing Technologies (PET 2004)*, 2004.

[Bran 11] W. A. Brandi. *In Search of Search Privacy*. PhD thesis, University of Pretoria, 2011.

[Bras 87] G. Brassard, C. Crépeau, and J.-M. Robert. "All-or-nothing disclosure of secrets". In: *Advances in CryptologyÑCRYPTOÕ86*, pp. 234–238, Springer, 1987.

[C So 12] C. Soghoian. "The History of the Do Not Track Header". Slight Paranoia, February 2012.

[Cast 09] J. Castellà-Roca, A. Viejo, and J. Herrera-Joancomarti. "Preserving user's privacy in web search engines". *Computer Communications*, Vol. 32, No. 13–14, pp. 1541–1551, 2009.

[Cast 10] D. Castellà, H. Blanco, F. Giné, and F. Solsona. "Combining Hilbert SFC and bruijn graphs for searching computing markets in a P2P system". In: *Proceedings of the 16th international Euro-Par conference on Parallel processing: Part I*, pp. 471–483, Springer-Verlag, Berlin, Heidelberg, 2010.

[Cast 11a] D. Castella, F. Gine, F. Solsona, and J. L. Lerida. "A Resilient Architecture Oriented to P2P Computing". In: *Proceedings of the 2011 IEEE 10th International Symposium on Network Computing and Applications*, pp. 41–50, IEEE Computer Society, Washington, DC, USA, 2011.

[Cast 11b] D. Castellà, H. Blanco, F. Giné, and F. Solsona. "A computing resource discovery mechanism over a P2P tree topology". In: *Proceedings of the 9th international conference on High performance computing for computational science*, pp. 366–379, Springer-Verlag, Berlin, Heidelberg, 2011.

[Cast 13] S. Castillo-Pérez and J. García-Alfaro. "Onion routing circuit construction via latency graphs". *Computers & Security*, Vol. 37, No. 0, pp. 197 – 214, 2013.

[Cata 13] D. Catalano, M. Di Raimondo, D. Fiore, R. Gennaro, and O. Puglisi. "Fully non-interactive onion routing with forward secrecy". *International Journal of Information Security*, Vol. 12, No. 1, pp. 33–47, 2013.

[Chau 81] D. L. Chaum. "Untraceable electronic mail, return addresses, and digital pseudonyms". *Commun. ACM*, Vol. 24, No. 2, pp. 84–90, Feb. 1981.

[Chau 92] D. Chaum and T. Pedersen. "Wallet databases with observers". In: L. N. in Computer Science, Ed., *Advances in Cryptology – CRYPTO'92*, pp. 89–105, 1992.

[Chor 97] B. Chor, N. Gilboa, and M. Naor. "Private Information Retrieval by Keywords". 1997.

[Chor 98] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. "Private information retrieval". *Journal of the ACM*, Vol. 45, No. 6, pp. 965–981, 1998.

[Chow 09] R. Chow and P. Golle. "Faking contextual data for fun, profit, and privacy". In: *Proceedings of the 8th ACM workshop on Privacy in the electronic society – WPES'09*, pp. 105–108, 2009.

[Coop 08] A. Cooper. "A survey of query log privacy-enhancing techniques from a policy perspective". *ACM Transactions Web*, Vol. 2, No. 4, pp. 19:1–19:27, Oct. 2008.

[Cove 12] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[Cran 06] L. F. Cranor, P. Guduru, and M. Arjula. "User interfaces for privacy agents". *ACM Transactions Computer-Human Interaction*, Vol. 13, No. 2, pp. 135–178, June 2006.

[Cran 12] L. F. Cranor. "Necessary But Not Sufficient: Standardized Mechanisms for Privacy Notice and Choice". *Journal of Telecommunications and High Technology Law*, Vol. 10, No. 2, pp. 273–445, Nov. 2012.

[Dane 09] G. Danezis, C. Diaz, and P. Syverson. "Systems for anonymous communication". In: *CRC cryptography and network security series*, Chapman Hall/CRC, 2009.

[Daou 09] M. Daoud, L. Tamine-Lechani, and M. Boughanem. "Towards a graph-based user profile modeling for a session-based personalized search". *Knowledge Information Systems*, Vol. 21, No. 3, pp. 365–398, 2009.

[Deer 90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. "Indexing by latent semantic analysis". *"Journal of the American Society for Information Science"*, Vol. 41, No. 6, pp. 391–407, 1990.

[Desm 90] Y. Desmedt and Y. Frankel. "Threshold cryptosystems". In: L. N. in Computer Science, Ed., *Advances in Cryptology – CRYPTO'89*, pp. 307–315, 1990.

[Ding 04] R. Dingledine, N. Mathewson, and P. Syverson. "Tor: the second-generation onion router". In: *Proceedings of the 13th conference on USENIX Security Symposium*, pp. 21–31, 2004.

[Domi 09a] J. Domingo-Ferrer, M. Bras-Amorós, Q. Wu, and J. Manjón. "User-Private Information Retrieval Based on a Peer-to-Peer Community". *Data & Knowledge Engineering*, Vol. 68, No. 11, pp. 1237–1252, 2009.

[Domi 09b] J. Domingo-Ferrer, A. Solanas, and J. Castellà-Roca. "h(k)-Private Information Retrieval from Privacy-Uncooperative Queryable Databases". *Online Information Review*, Vol. 33, pp. 720–744, 2009.

[Dosh 13] N. Doshi and D. Jinwala. "AB-OR: Improving the Efficiency in Onion Routing Using Attribute Based Cryptography". In: *Computer Networks & Communications (NetCom)*, pp. 425–432, Springer New York, 2013.

[Duck 13] DuckDuckGo. https://duckduckgo.com/, 2013.

[E St 10] E. Steel. "A web pioneer profiles users by name". The Wall Street Journal, October 2010.

[Ecke 10] P. Eckersley. "How unique is your web browser?". In: *Proceedings of the 10th international conference on Privacy enhancing technologies*, pp. 1–18, Springer-Verlag, Berlin, Heidelberg, 2010.

[Eick 13] C. Eickhoff, K. Collins-Thompson, P. Bennett, and S. Dumais. "Designing human-readable user profiles for search evaluation". In: *Proceedings of the 35th European conference on Advances in Information Retrieval*, pp. 701–705, Springer-Verlag, Berlin, Heidelberg, 2013.

[Elec 00] Electronic Privacy Information Center (EPIC). "Pretty Poor Privacy: An Assessment of P3P and Internet Privacy". http://epic.org/reports/prettypoorprivacy.html, June 2000.

[ElGa 85] T. ElGamal. "A public-key cryptosystem and a signature scheme based on discrete logarithms". *IEEE Transactions on Information Theory*, Vol. 31, pp. 469–472, 1985.

[Erol 11a] A. Erola, J. Castellà-Roca, G. Navarro-Arribas, and V. Torra. "Semantic microaggregation for the anonymization of query logs using the open directory project". *SORT-Statistics and Operations Research Transactions*, Vol. 0, pp. 41–58, 2011.

[Erol 11b] A. Erola, J. Castellà-Roca, A. Viejo, and J. M. Mateo-Sanz. "Exploiting social networks to provide privacy in personalized web search". *Journal of Systems and Software*, Vol. 84, No. 9, pp. 1734–1745, 2011.

[Foxt 13] Foxtor. http://cups.cs.cmu.edu/foxtor/, 2013.

[Gold 99] D. Goldschlag, M. Reed, and P. Syverson. "Onion Routing for Anonymous and Private Internet Connections". *Communications of the ACM*, Vol. 42, pp. 39–41, 1999.

[Goog 12] Google Official History and ComScore. "Google Annual Search Statistics". http://www.statisticbrain.com/google-searches/, 2012.

[Goog 13] Google Privacy Center. http://www.google.com/privacy, 2013.

[Grub 69] F. E. Grubbs. "Procedures for Detecting Outlying Observations in Samples". *Technometrics*, Vol. 11, No. 1, pp. 1–21, 1969.

[He 00] D. He and A. Göker. "Detecting session boundaries from web user logs". In: *Proceedings of the BCS-IRSG 22nd annual colloquium on information retrieval research*, pp. 57–66, 2000.

[Hoch 02] H. Hochheiser. "The platform for privacy preference as a social protocol: An examination within the U.S. policy context". *ACM Transactions Internet Technology*, Vol. 2, No. 4, pp. 276–306, Nov. 2002.

[Huan 04] X. Huang, F. Peng, A. An, and D. Schuurmans. "Dynamic Web Log Session Identification with Statistical Language Models". *Journal of the American Society for Information Science and Technology*, Vol. 55, pp. 1290–1303, 2004.

[I2P 13] I2P. "Invisible Internet Project". http://www.i2p2.de/index.html, 2013.

[iPro 08] iProspect.com. "iProspect Blended Search Results Study". *http://www.iprospect.com*, 2008.

[Ixqu 13] Ixquick. https://www.ixquick.com/, 2013.

[Jaga 05] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. "BATON: a balanced tree structure for peer-to-peer networks". In: *Proceedings of the 31st international conference on Very large data bases*, pp. 661–672, VLDB Endowment, 2005.

[Jone 08] 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*, pp. 853–862, ACM, 2008.

[K Ha 06] K. Hafner and M. Richtel. "Google Resists U.S. Subpoena of Search Data". New York Times, January 2006.

[Kaas 03] M. F. Kaashoek and D. R. Karger. "Koorde: A Simple Degree-Optimal Distributed Hash Table". In: *International Conference on Peer-to-Peer Systems*, pp. 98–107, 2003.

[Kram 13] T. Kramár, M. Barla, and M. Bieliková. "Personalizing search using socially enhanced interest model, built from the stream of user's activity". *Journal Web Engineering*, Vol. 12, No. 1-2, pp. 65–92, Feb. 2013.

[Kush 97] E. Kushilevitz and R. Ostrovsky. “Replication is not needed: single database, computationally-private information retrieval”. In: *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science*, pp. 364–373, IEEE press, 1997.

[Leun 12] K. W.-T. Leung, D. L. Lee, W. Ng, and H. Y. Fung. “A framework for personalizing web search with concept-based user profiles”. *ACM Transactions Internet Technology*, Vol. 11, No. 4, pp. 17:1–17:29, March 2012.

[Levi 02] B. N. Levine and C. Shields. “Hordes: A Multicast Based Protocol for Anonymity”. *Journal of Computer Security*, Vol. 10, pp. 213–240, 2002.

[Lind 10] Y. Lindell and E. Waisbard. “Private web search with malicious adversaries”. In: *Proceedings of the 10th international conference on Privacy enhancing technologies – PETS’10*, pp. 220–235, 2010.

[Logu 03] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. “Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience”. In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 395–406, ACM, New York, NY, USA, 2003.

[Lua 05] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. “A Survey and Comparison of Peer-to-Peer Overlay Network Schemes”. *IEEE Communications Surveys and Tutorials*, Vol. 7, pp. 72–93, 2005.

[M Ba 05] M. Barbaro and T. Zeller. “A Face is Exposed for AOL Searcher No. 4417749”. New York Times, August 2005.

[M Ja 99] M. Jakobsson and A. Juels. “Millimix: mixing in small batches”. DIMACS Technical report 99-33, 1999.

[M Ka 06] M. Kamvar and S. Baluja. “A large scale study of wireless search behavior: Google mobile search”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 701–709, 2006.

[M Ra 06] M. Rasch. “Google’s data minefield: Search engine vs Government”. The Register, http://www.theregister.co.uk/, 2006.

[Malk 02] D. Malkhi, M. Naor, and D. Ratajczak. "Viceroy: A Scalable and Dynamic Emulation of the Butterfly". In: ACM, Ed., *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pp. 183–192, 2002.

[Mann 99] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA, 1999.

[Maye 11] J. Mayer, A. Narayanan, and S. Stamm. "Do not track: A universal third-party web tracking opt out". *IETF Request for Comments*, pp. 1–12, 2011.

[Maym 02] P. Maymounkov and D. Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". In: *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pp. 53–65, Springer-Verlag, London, UK, UK, 2002.

[Mesh 08] E. Meshkova, J. Riihijärvi, M. Petrova, and P. Mähönen. "A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks". *Comput. Netw.*, Vol. 52, No. 11, pp. 2097–2128, Aug. 2008.

[Moon 96] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz. "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve". *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, p. 2001, 1996.

[Muru 09] M. Murugesan and C. Clifton. "Providing Privacy through Plausibly Deniable Search". In: *In SDM*, 2009.

[Nati 10] National Institute of Standards and Technologies (NIST). "Guide to Protecting the Confidentiality of Personally Identifiable Information". Special Publication 800-122, April 2010.

[Netc 12] Netcraft. "Anti-Phishing and PCI Security Services, September 2012 Web Server Survey". *http://news.netcraft.com/archives/category/web-server-survey/*, september 2012.

[NIST 07] NIST. "Recommendation for Key Management". Special Publication 800–57 Part 1, 2007.

[ODP 13] ODP. "Open Directory Project". http://www.dmoz.org/, 2013.

[Oliv 03] M. S. Olivier. "A layered architecture for privacy-enhancing technologies". *South African Computer Journal*, Vol. 31, pp. 53–61, 2003.

[Oliv 04] M. S. Olivier. "Privacy under Conditions of Concurrent Interaction with Multiple Parties". In: *Data and Applications Security XVII — Status and Prospects*, pp. 105–118, 2004.

[Ostr 07] R. Ostrovsky and W. E. Skeith-III. "A survey of single-database pir: techniques and applications". In: *Lecture Notes in Computer Science*, pp. 393–411, 2007.

[Page 13] Page Wash. http://www.pagewash.com/, 2013.

[Pedd 10] S. T. Peddinti and N. Saxena. "On the privacy of web search based on query obfuscation: a case study of TrackMeNot". In: *Proceedings of the 10th international conference on Privacy enhancing technologies – PETS'10*, pp. 19–37, 2010.

[Pfit 01] A. Pfitzmann and M. Köhntopp. "Anonymity, unobservability, and pseudeonymity &#8212; a proposal for terminology". In: *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability*, pp. 1–9, Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[Plax 97] C. G. Plaxton, R. Rajaraman, and A. W. Richa. "Accessing nearby copies of replicated objects in a distributed environment". In: *Proceedings of the ninth annual ACM symposium on Parallel algorithms and architectures*, pp. 311–320, 1997.

[Ratn 01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. "A scalable content-addressable network". *SIGCOMM Comput. Commun. Rev.*, Vol. 31, No. 4, pp. 161–172, Aug. 2001.

[Reay 09] I. Reay, S. Dick, and J. Miller. "A large-scale empirical study of P3P privacy policies: Stated actions vs. legal obligations". *ACM Transactions Web*, Vol. 3, No. 2, pp. 6:1–6:34, Apr. 2009.

[Rebo 10] D. Rebollo-Monedero and J. Forné. "Optimized query forgery for private information retrieval". *IEEE Transactions Information Theory*, Vol. 56, No. 9, pp. 4631–4642, Sep. 2010.

[Reit 98] M. Reiter and A. Rubin. "Crowds: anonymity for Web transactions". *ACM Transactions on Information and System Security*, Vol. 1, No. 1, pp. 66–92, 1998.

[Rows 01] A. Rowstron and P. Druschel. "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems". *IN: MIDDLEWARE*, pp. 329–350, 2001.

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

[Sanc 13] D. Sánchez, J. Castellà-Roca, and A. Viejo. "Knowledge-based scheme to create privacy-preserving but semantically-related queries for web search engines". *Inf. Sci.*, Vol. 218, pp. 17–30, Jan. 2013.

[Sarm 07] S. Sarmady. "A Survey on Peer-to-Peer and DHT". 2007.

[Schn 91] C. P. Schnorr. "Efficient Signature Generation by Smart Cards". *Journal of Cryptology*, Vol. 4, pp. 161–174, 1991.

[Shen 05] X. Shen, B. Tan, and C. Zhai. "Ucair: Capturing and exploiting context for personalized search". In: *Proceedings of the ACM SIGIR 2005 Workshop on Information Retrieval in Context (IRiX)*, p. 45, Citeseer, 2005.

[Soo 02] W. H. Soo, A. Samsudin, and A. Goh. "Efficient Mental Card Shuffling via Optimised Arbitrary-Sized Benes Permutation Network". In: L. N. in Computer Science, Ed., *Proceedings of the 5th International Conference – ISC 2002*, pp. 446–458, 2002.

[Star 13] Start Page. https://startpage.com/, 2013.

[Stat 13] Statista. "Average number of search queries per user in European countries in December 2012". http://www.statista.com/statistics/254710/average-number-of-search-queries-per-user-in-european-countries/, 2013.

[Stoi 01] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications". In: *ACM SIGCOMM Computer Communication Review. Vol. 31. No. 4*, pp. 149–160, SIGCOMM 2001, August 2001.

[Swee 02] L. Sweeney. "k-anonymity: a model for protecting privacy". *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, Vol. 10, No. 5, pp. 557–570, Oct. 2002.

[Syve 11] P. Syverson. "Practical vulnerabilities of the Tor anonymity network". In: *Advances in Cyber Security: Technology, Operation and Experiences*, 2011.

[Tene 12] O. Tene and J. Polenetsky. "To Track or Do Not Track: Advancing Transparency and Individual Control in Online Behavioral Advertising". *Minn. JL Sci. & Tech.*, Vol. 13, p. 281, 2012.

[Tor 13] Tor. "The Tor Project". http://www.torproject.org/, 2013.

[Torb 13] Torbutton. http://freehaven.net/∼squires/torbutton/, 2013.

[Trac 13] TrackMeNot. "TMN". http://mrl.nyu.edu/dhowe/trackmenot, 2013.

[Tsai 09] J. Tsai, S. Egelman, L. Cranor, and A. Acquisti. "The impact of privacy indicators on search engine browsing patterns". In: *Proceedings of the 5th Symposium on Usable Privacy and Security*, pp. 29:1–29:1, ACM, New York, NY, USA, 2009.

[Twit 13] Twitter. https://twitter.com/, 2013.

[Ucai 05] Ucair. "UCAIR Personalized Search Toolbar". http://ucair.wordpress.com/2005/09/09/ucair-personalized-search-toolbar/, 2005.

[Univ 48] "Universal Declaration of Human Rights". http://www.un.org/Overview/rights.html, 1948.

[Viej 10] A. Viejo and J. Castellà-Roca. "Using social networks to distort users' profiles generated by web search engines". *Computer Networks*, Vol. 54, No. 9, pp. 1343–1357, 2010.

[Viej 12a] A. Viejo, J. Castellà-Roca, O. Bernado, and J. M. Mateo-Sanz. "Single-party private web search". In: *Proceedings of the 2012 Tenth Annual International Conference on Privacy, Security and Trust (PST)*, pp. 1–8, IEEE Computer Society, Washington, DC, USA, 2012.

[Viej 12b] A. Viejo, D. Sánchez, and J. Castellà-Roca. "Using profiling techniques to protect the user's privacy in twitter". In: *Proceedings of the 9th international conference on Modeling Decisions for Artificial Intelligence*, pp. 161–172, Springer-Verlag, Berlin, Heidelberg, 2012.

[Viej 13] A. Viejo and D. Sánchez. "Providing Useful and Private Web Search by Means of Social Network Profiling". In: *Proceedings of the 2013 Eleventh Annual International Conference on Privacy, Security and Trust (PST)*, To appear, 2013.

[West 10] B. Westermann, R. Wendolsky, L. Pimenidis, and D. Kesdogan. "Cryptographic protocol analysis of AN.ON". In: *Proceedings of the 14th international conference on Financial Cryptography and Data Security*, pp. 114–128, Springer-Verlag, Berlin, Heidelberg, 2010.

[Wrig 04] M. K. Wright, M. Adler, B. N. Levine, and C. Shields. "The predecessor attack: An analysis of a threat to anonymous communications systems". *ACM Transactions on Information and System Security*, Vol. 7, No. 4, pp. 489ñ–522, 2004.

[Xu 07] Y. Xu, B. Zhang, and K. Wang. "Privacy-enhancing personalized web search". In: *Proceedings of the 16th international conference on World Wide Web*, pp. 591–600, ACM Press, 2007.

[Ye 09] S. Ye, F. Wu, R. Pandey, and H. Chen. "Noise Injection for Search Privacy Protection". In: *Proceedings of the 2009 International Conference on Computational Science and Engineering - Volume 03*, pp. 1–8, IEEE Computer Society, Washington, DC, USA, 2009.

[Yipp 13] Yippy. http://yippy.com/, 2013.

[Zhao 01] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location". Tech. Rep., Techreport, Berkeley, CA, USA, 2001.