Preserving User's Privacy in Web Search Engines

Jordi Castellà-Roca [a], Alexandre Viejo [b,a], Jordi Herrera-Joancomartí [c,d]

[a] *Rovira i Virgili University, Dept. of Computer Engineering and Maths, UNESCO Chair in Data Privacy, Av. Països Catalans 26, E-43007 Tarragona, Spain*
E-mail: {jordi.castella, alexandre.viejo}@urv.cat

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

[c] *Departament d'Enginyeria de la Informació i les Comunicacions, Universitat Autònoma de Barcelona, Edifici Q, Campus de Bellaterra, 08193 Bellaterra, Spain.*
E-mail: jherrera@deic.uab.cat

[d] *Internet Interdisciplinari Institute (IN3), Universitat Oberta de Catalunya, Spain.*
E-mail: jherreraj@uoc.edu

Abstract

Web search engines (*e.g.* Google, Yahoo, Microsoft Live Search...) are widely used to find certain data among a huge amount of information in a minimal amount of time. However, these useful tools also pose a privacy threat to the users: web search engines profile their users by storing and analyzing past searches submitted by them. To address this privacy threat, current solutions propose new mechanisms that introduce a high cost in terms of computation and communication. In this paper we present a novel protocol specially designed to protect the users' privacy in front of web search profiling. Our system provides a distorted user profile to the web search engine. We offer implementation details and computational and communication results that show that the proposed protocol improves the existing solutions in terms of query delay. Our scheme provides an affordable overhead while offering privacy benefits to the users.

Key words: Cryptography, Privacy, Private information retrieval, Web search

1 Introduction

Communication networks enable us to reach a very large volume of information in a minimal amount of time. Furthermore, that huge quantity of data can be accessed at any time and any place with a capable device (*e.g.* a laptop, a PDA...) and an Internet connection. Nowadays, it is pretty common to access easily to both resources. In the future, it will be even easier.

However, useful information about a specific topic is hidden among all the available data and it can be really challenging to find it since that information can be scattered around the Word Wide Web.

Web search engines (*e.g.* Google, Yahoo, Microsoft Live Search...) are widely used to do this hard job for us. The 84% of the Internet users have used a web search engine at least once. For the 32%, web search engines are an essential tool to address their everyday duties [1]. Among the different search engines, Google is the most used in the U.S. with a 43.7% of the total amount of searches performed in 2006 [2]. Google improves its performance (it gives personalized search results) by storing a record of visited sites and past searches submitted by each user [3] (Web History).

Those searches can reveal a lot of information from individual users or the institutions they work for. For example, let us imagine an employee of a certain company A. This employee uses Google to obtain information about a certain technology. If a company B, which is a direct competitor of A, knows this situation, it can infer that this technology will be used in the new products offered by A. This knowledge gives to B an important advantage over A. Another example of this situation occurs when a person is applying for a certain job. In this case, if the employer knows that the applicant has been looking for information regarding a certain disease, she can use this knowledge to choose another person for the job. In both examples, the *attacker* (the entity who gets some advantage over the other) benefits from the lack of a privacy-preserving mechanism between the user and the web search engine.

1.1 Previous work

The problem of submitting a query to a web search engine while preserving the users' privacy can be seen as a Private Information Retrieval (PIR) problem. In a PIR protocol, a user can retrieve a certain value from a database while the server, which holds the database, gets no knowledge about the data requested by the user. In our case, the server is represented by the web search engine and the database is represented by the web pages that the web search engine stores.

The first PIR protocol was designed by Chor, Goldreich, Kushilevitz and Sudan [4,5]. Their scheme is based on several servers holding the same database. These servers cannot communicate between them. The main shortcoming of this proposal is that it is unable to work with only one server (single-database PIR), which is the web search engine in our scenario. Also, it is not realistic to assume that servers are unable to communicate between them.

Single-database PIR protocols can be found in the literature. The first one was presented in [6] (see [7] for a detailed survey on single-database PIR developments). These schemes are more suitable for working with web search engines. However, they suffer from some fundamental problems that make their use unfeasible in communications between a user and a web search engine. The main shortcomings are the following:

(1) Single-database PIR schemes are not suited for large databases. In 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 this 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 to all records in the database. Note that if a certain user only accesses to 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. This situation is not realistic because the database is not managed by the user. In [8], 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 is not feasible in a real scenario since web search engines do not use this model.

(3) Finally, it is assumed that the server which holds the database collaborates with the user in the PIR protocol. Nevertheless, this assumption is not realistic because the server has no motivation to protect the privacy of the users. In fact, the use of a PIR protocol is a disadvantage for the server since this limits its profiling ability. According to that, the user should take care of her own privacy by herself. She should not expect any collaboration from the web search engine.

Another approach to provide privacy in web searches is the use of a general purpose anonymous web browsing mechanism. Simple mechanisms to achieve a certain level of anonymity in web browsing include: (i) the use of proxies; or (ii) the use of dynamic IP addresses (for instance, using DHCP). Proxies do not solve the privacy problem. This solution only moves the privacy threat from the web search engine to the proxies themselves. A proxy will prevent the web search engine from profiling the users, but the proxy will be able to profile them instead. Regarding the use of dynamic IP addresses, this option has the following drawbacks:

- The renewal policy of the dynamic IP address is not controlled by the user but the network operator. This operator can always give the same IP address to the same Media Access Control (MAC) address.
- Certain users require static IP addresses.

More reliable systems are those based on the cryptographic approach of onion routing. The Tor anonymity network [9] is an example of that. However, Tor cannot be installed and configured straightforwardly. This is a problem because a wrong configuration may allow some reported attacks. In addition to that, one of the most important drawbacks of this scheme is the fact that the HTTP requests over the Tor network are fairly slow [10]. Thus, they introduce important delays in the web search process.

In addition to PIR protocols and anonymous web browsing, some proposals for the specific problem of private web search have been made in the last years. The main ones are next reviewed.

Authors in [11] distinguish and define four levels of privacy protection in personalized web search. In the first level (pseudo identity), user identity is replaced with a pseudo identity that contains less personal information. This level is not enough to protect the user's privacy because the information from a certain user can be aggregated to facilitate her posterior identification. The AOL [12] scandal is an example of this. The second level provides a single identity for a group of users (group identity). The search engine can only build a group profile for the group of users instead of a single profile for each user. In the third level, the user identity is not available to the search engine (no identity). For example, users send queries through an anonymous network. Finally, in the fourth level, neither the user identity nor the query are available to the search engine. Although [11] does not propose any implementable solution (it only offers theoretical models), authors point out that this fourth level of privacy protection may have the highest cost due to large communication and cryptographic costs.

A more practical approach can be found in [10]. In this work, authors propose a tool named Private Web Search (PWS). This tool falls in the third level of privacy protection according to the classification in [11]. PWS is a Firefox plugin with HTTP proxy capabilities that is used to protect the users' privacy. This plugin is a client for the Tor anonymity network. Search queries are routed through the HTTP proxy and the Tor network. In this scheme, the cost of submitting a query to Google is about 10 second on average, i.e. 33

times slower than a direct connection (0.3 seconds). Furthermore, it is worth to mention that this delay is obtained using paths of length 2, instead of the default length of 3.

In [13], authors present a system to automatically build a hierarchical user profile using browsing history and past emails. This profile represents the user's implicit personal interests. In this profile, general terms with higher frequency are at higher levels and specific terms with lower frequency are at lower levels. The user specifies a threshold, which represents the minimum number of documents where a frequent term is required to occur. Next, a new partial profile is build using the threshold and the complete profile, this partial profile has the highest levels. The authors developed a search engine wrapper on the server side that incorporates the partial user profile with the results retrieved from the web search engine. This scheme was tested using the MSN search engine. Nonetheless, this solution does not prevent the search engine from building user's profiles because users submit their queries directly to the search engine.

1.2 Contribution and plan of this paper

We propose a novel protocol, the Useless User Profile (UUP) protocol, specially designed to protect the users' privacy in front of web search profiling. Our system provides a distorted user profile to the web search engine. The proposed protocol submits standard queries to the web search engine. Thus, it does not require any change in the server side. In addition to that, this scheme does not need the server to collaborate with the user. The computational cost and communication overhead obtained in our tests prove that this new protocol improves the performance of existing proposals.

The paper is organized as follows. In Section 2 some cryptographic background is provided. Section 3 describes the Useless User Profile (UUP) protocol and offers a security analysis. Implementation details and performance results are given in Section 4. Finally, Section 5 concludes the paper and gives some ideas for further research.

2 Cryptographic building blocs

In this section, we outline the cryptographic tools our protocol is based on.

2.1 *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 [14], n users share a public key y and the corresponding unknown private 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.

2.1.1 *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 $\alpha = \alpha_1 + \ldots + \alpha_n$.

2.1.2 *Message encryption*

Message encryption can be performed using the standard ElGamal encryption function [15]. Given a message m and a public key y, a random value r is generated and the ciphertext is computed as follows:

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

2.1.3 *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 [16].

2.2 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 [17]:

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

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 Useless user profile protocol for web search queries

In this section we present our Useless User Profile (UUP) protocol that allows users to submit queries to a web search engine without disclosing any useful personal information. In this way, this scheme prevents web search engines from obtaining valid profiles from the users.

First of all, we describe the entities involved in our scheme. Next, we give a general view of the proposed protocol and we enumerate the privacy properties of the proposed scheme. In the last subsection, a detailed description of the UUP protocol is given.

3.1 Entities

Our scenario contains the following entities:

- *Users (U).* They are the individuals who submit queries to the web search engine. Their motivation is to protect their own privacy.
- *The central node (C).* It groups users in order to execute the UUP protocol that is next introduced. The main objective of this entity is to get in touch all the users that want to submit a query.
- *The web search engine (W).* It is the server that holds the database. It can be Google, Yahoo or Microsoft Live Search among others. It has no motivation to preserve users privacy.

3.2 Protocol overview

The main idea of our scheme is that each user who wants to submit a query will not send her own query but a query of another user instead. At the same time, her query is submitted by another user. Regarding privacy concerns, the relevant point is that users do not know which query belongs to each user. This is achieved using cryptographic tools. As a result of this process, each user submits very different kinds of queries. Those queries are not liable to a certain person since each user submit queries which are not generated by herself. Using this approach, the web search engine cannot generate a real profile of a certain individual.

The proposed scheme, at a high level, works as follows: the central node groups n users who want to submit a query. Those n users execute an anonymous query retrieval protocol where each user U_i, for $i \in \{1, \dots, n\}$, gets a query from one of the other $n-1$ users. U_i does not know the source of the received query. To achieve this goal, all queries are first shuffled and then distributed. This process is performed using encryption, re-masking and permutation by means of a multi-party protocol executed by all the users to ensure its fairness. Then, each user submits the received query to the web search engine. The answer from the search engine is broadcast to all the group members. Each user takes only her answer. The remaining answers are discarded.

In order to ensure the correctness of this process, our protocol assumes that the users follow the protocol and that there are no collusion between two entities of the system. These assumptions are reasonable since the protocol's main objective is to avoid the web search profiling done by the web search engine.

Regarding the privacy requirements of the users, our scheme fulfills the following properties:

- Users must not link a certain query with the user who has generated it.
- The central node must not link a certain query with the user who has generated it.
- The web search engine must be unable to construct a reliable profile of a certain user.

3.3 Protocol description

The UUP protocol is composed of four subprotocols:

- Group set up.

- Group key generation.
- Anonymous query retrieval.
- Query submission and retrieval.

3.3.1 Group set up

When a user U_i wants to submit a query to the web search engine, she sends a message to the central node C requesting to be included in a group.

The central node C receives all user requests. Once it has n requests, it creates a new user group $\{U_1, \ldots, U_n\}$ and notifies the n users that they belong to the same group. After this first step, users in this group can establish a communication channel between them. In this way, each user can send messages to the rest of the group directly. The central node is no longer needed.

3.3.2 Group key generation

The UUP protocol uses a group key which is generated using a fair threshold encryption scheme. All users follow these steps:

(1) Users $\{U_1, \ldots, U_n\}$ agree on a large prime p where $p = 2q + 1$ and q is prime too. Next, they pick an element $g \in \mathbb{Z}_p^*$ of order q.
(2) Users $\{U_1, \ldots, U_n\}$ generate an ElGamal single public key y using the n-out-of-n threshold ElGamal encryption described in Section 2.1.1. Each user U_i keeps her private key α_i secret.

3.3.3 Anonymous query retrieval

Each group user $\{U_1, \ldots, U_n\}$ submits her personal query m_i secretly and obtains one query m^i from another group member anonymously by executing the following protocol:

(1) For $i = 1, \ldots, n$, each user U_i does the following steps:
    (a) U_i generates a random value r_i and encrypts her query m_i using the group key y. The result of this operation is:

$$E_y(m_i, r_i) = (c1_i, c2_i) = c_i^0$$

    (b) U_i sends c_i^0 to the other group members U_j, for $\forall j \neq i$.
(2) Assuming a predefined order for the users in the group (from 1 to n), we denote as $\{c_1^0, \ldots, c_n^0\}$ the ordered cryptograms that each user U_i owns at the end of the sending process. Then, for $i = 1, \ldots, n$, each user U_i performs the following operations:

(a) U_i re-masks the cryptograms $\{c_1^{i-1}, \ldots, c_n^{i-1}\}$ received from U_{i-1} and she obtains a re-encrypted version $\{e_1^{i-1}, \ldots, e_n^{i-1}\}$ using the re-masking algorithm defined in section 2.2. Note that U_1 re-masks the cryptograms $\{c_1^0, \ldots, c_n^0\}$ that she already owns. The resulting ciphertexts correspond to the same initial cleartexts.

(b) U_i permutes the order of the cryptograms at random and obtains a reordered version $\{c_1^i, \ldots, c_n^i\} = \{e_{\sigma(1)}^{i-1}, \ldots, e_{\sigma(n)}^{i-1}\}$.

(c) U_i sends $\{c_1^i, \ldots, c_n^i\}$ to U_{i+1}. User U_n broadcasts $\{c_1^n, \ldots, c_n^n\}$ to all the group members.

(3) Let us denote $\{c_1, \ldots, c_n\} = \{c_1^n, \ldots, c_n^n\}$. At this step, each user U_i has 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, probably m^i does not 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 by computing:

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

3.3.4 Query submission and retrieval

(1) Each group member U_i submits the retrieved m^i to the web search engine.
(2) Upon receiving the corresponding answer a^i from the web search engine, each user broadcasts it to the rest of the group members.
(3) Each user takes the answer a^i that corresponds to her original query from all the received answers.

3.4 Security analysis

Our proposal has been designed to preserve the privacy of the users when they submit queries to a web search engine. According to that, a successful attacker would be able to link a certain query to the user who has generated it.

We consider that the computational power of an attacker does not allow him to break current computationally secure cryptosystems. Also, as explained in Section 3.2, our protocol assumes that the users follow the proposed protocol correctly and that there are no collusions between entities.

The attacker can be any entity (one of the three entities of the protocol or an external one). However, external attackers can get, at most, the same information that an internal entity. For that reason, we perform our analysis assuming that the attacker is an internal entity.

3.4.1 Dishonest user

A dishonest user U_a who is part of a group follows all the steps of the presented protocol. At the end of the process, U_a owns the original ciphertexts $\{c_1^0, \ldots, c_n^0\}$ (before the re-masking/permuting step) which contain the queries from all the users of the group. If U_a would be able to decrypt these ciphertexts, she would know which query has sent each user. Nevertheless, as long as the secret keys $(\alpha_1, \ldots, \alpha_n)$ are not compromised, she is unable to do it. In this way, the proposed system preserves the privacy of the users.

3.4.2 Dishonest central node

The central node C only participates in the first step of the protocol (Section 3.3.1). Its purpose is to receive requests from users who want to form a group. Upon C receives n requests from n different users, the corresponding group is formed and C is no longer needed. According to that, C cannot get any posterior transmission between the users, hence it cannot link any query to any user.

3.4.3 Dishonest web search engine

The web search engine W participates in the last step of the protocol (Section 3.3.4). This entity receives all the queries from the users who form the group and answers them. In this way, W can link a certain query m^i with the user U_i who has submitted it. However, W alone cannot determine whether m^i belongs to U_i or whether it has been generated by another user. Therefore, W has a useless profile of U_i.

4 Implementation details and experimental results

The protocol described in Section 3 prevents a search engine from obtaining a reliable profile of a certain user, i.e. it brings a higher degree of privacy to the users. From the strictly technical point of view, the cost of achieving this privacy degree can be measured in terms of query delay. The proposed protocol requires some cryptographic operations and network communications that increase this delay.

We have implemented our protocol and we next present some results regarding its performance in a practical scenario. These results prove that our proposal introduces a delay which can be assumed by the user.

4.1 Implementation and configuration details

The proposed system requires two components: the central node and the client application (see Section 3.1). These two components have been implemented using the Java programming language [18]. This allows application portability.

The central node C is a process (daemon) that listens to client requests in a fixed TCP port. After receiving n requests, C creates a new group and sends a message with the users' IP address and the number of the port to be used. In order to improve the protocol's performance, this message also contains the large prime p, and the $g \in Z_p^*$ element (see section 3). This reduces the number of messages that must be transmitted. The configuration of the central node includes the number n of users needed to form a group, the port number, and the length of the large prime p.

The client application is a java applet which is accessed by an html web page (see Figure 1) that allows users to make a search in a transparent manner.

Fig. 1. Client interface

The client interface displays a form field (similar to the one that can be found in a classic web search engine) where the user must type her query. The search process is started once the *search* button is pressed. The text is sent from the form field to the applet using a Javascript code. The applet runs the proposed UUP protocol establishing connections with all the group members and with the web search engine. Finally, it shows the result to the user.

All communications between entities are performed using TCP connections. Messages are implemented using the XML format.

The UUP protocol can be configured with two different parameters: the group size n and the key length l used by the cryptographic operations.

Obviously, the larger n is, the better the algorithm hides the real profile of each user. However, when a user wants to submit a query, she has to wait until another $n-1$ users want to do the same. Thus, when n increases, the waiting time needed to create a group also increases. In the same way, a larger group implies more messages between the group members and therefore a higher delay due to the communication.

The key length l refers to the size (in bits) of the ElGamal cryptosystem (see section 2.1) used in the UUP protocol. A short key is considered not secure, hence it is required a minimum key length. On the other hand, the time consumed in the cryptographic operations (key generation, encryption, decryption and re-masking) of the UUP protocol is directly related to the key size used.

The UUP protocol should use a large n and l to hide the real users' profiles and to offer enough security. At the same time, n and l should be short in order to introduce a low delay in the response. According to that, there is a trade-off for both parameters. In order to tune our system and to find the values of n and l that offer a reasonable combination of privacy, security and usability, we have used two testing environments. These are a controlled environment and an open environment (see Section 4.2.1).

The controlled environment is a Local Area Network (LAN). The time obtained by the UUP protocol in this environment is not affected by external factors. Thus, it allow us to evaluate the protocol behaviour in optimal conditions and perform a tuning analysis of the values n and l. In this environment, for each key length, we have run the protocol with different group sizes to see how both parameters affect the query delay. The group size n and the key length l that have been chosen are described in Section 4.3.1.

The results obtained in the tuning analysis within the controlled environment show the maximum n and l values which can be used to get a low query delay. In this environment, the protocol is evaluated in optimal conditions. Therefore, any combination of n and l that results in a high query delay is very likely to be unaffordable in the open environment (Internet).

According to that, the controlled environment provides the best configurations. Then, these configurations are tested in the open environment in order to get the *real* delay introduced by our proposal.

4.2.1 Equipment properties

The controlled environment refers to the execution of tests in a local area network. The network that was used in our tests was connected to the University network in order to get access to the web search engine. Nonetheless, all the internal network traffic came only from nodes involved in our tests.

All computers (central node and clients) share the same configuration (see Table 1). This uniformity allows us to get an average cost of the operations performed with these settings. Note that each computer executes only a single client application in each test.

Table 1
Controlled environment: equipment

Computers	CPU	Intel Core 2 CPU 6320 at 1.86GHz
	RAM	2 GBytes
	O.S.	Microsoft Windows Server 2003
	Java version	Java 6 Update 5
	Ethernet	100 Mbits
Network	Switch 100 Mbits	

In the open environment (see section 4.4), all the computers are located in different cities, hence they are connected between them through the Internet. Unlike the controlled environment, in this case all the computers have different specifications and use different technologies (and speeds) to connect to the Internet (see Table 2).

Table 2
Open environment: equipment

	Server	MacBook Air	Thinkpad X61s	iMac	PC
Specifications					
CPU	Intel Pentium 4 at 2.8 GHz	Intel Core 2 Duo at 1.6 GHz	Intel Core 2 Duo at 1.6 GHz	Intel Core 2 Duo at 2.0 GHz	Intel Pentium D at 2.8 GHz
RAM	1 GByte	2 GBytes	2 GBytes	2 GBytes	1 Gbyte
O.S.	Debian GNU/Linux	Mac OS X Leopard	MS Windows Vista Business	Mac OS X Leopard	MS Windows Vista Home
Java version	Java 6 Update 1	Java 6 Update 7	Java 6 Update 7	Java 6 Update 7	Java 6 Update 7
Internet connection					
Technology	Gigabit Ethernet	DSL	DSL	Cable	DSL
Download	100 Mbits	6 Mbits	3 Mbits	6 Mbits	8 Mbits
Upload	100 Mbits	500 Kbits	300 Kbits	300 Kbits	750 Kbits
Connection	Direct	Router	Bridge	Router	Router
Address	public	private DNAT	public	private DNAT	private DNAT

Like in the controlled environment, each computer executes only a single client application in each test.

4.2.2 Time measures

Sections 4.3.3 and 4.4.2 provide aggregated results in milliseconds that show the delay introduced by our scheme when submitting a single query. In addition to that, Sections 4.3.2 and 4.4.2 offer a detailed analysis of the time spent on each protocol step. This is done to show which steps are the most sensitive ones. Below, there is a description of each time interval:

- t_0: time interval required to initialize the applet.
- t_1: time interval 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 (see section 3.3.2).
- 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 send her key share $y_i = g^{\alpha_i}$ to the group members.
- 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 to 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 the user U_i sends her ciphertext c_i^0 until the last user broadcasts the ciphertexts $\{c_1, \cdots, c_n\}$ to all the group members (see step 2 of the protocol defined in section 3.3.3). 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 send the shares to the group members.
- 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 web search engine 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 proposed granularity of the algorithm allows us to determine the following periods:

- T_C: time interval required to perform cryptographic operations.

$$T_C = t_2 + t_5 + t_6 + t_8 + t_{11} + t_{13}$$

- T_N: time interval required to perform network operations.

$$T_N = t_1 + t_3 + t_4 + t_7 + t_9 + t_{10} + t_{12} + t_{14} + t_{15}$$

Note that t_{10} has been included as network time, although it includes the cryptographic operations of other users and their communications.

4.3 Controlled testing environment

In the controlled environment, there is no traffic from external sources. Therefore, the resulting delay can only be attributed to the proposed protocol.

In Section 4.3.1, we justify the selection of parameters n and l which has been used in our tests. In Section 4.3.2, we provide a detailed analysis of the time spent on each protocol step. Section 4.3.3 shows the time delay experienced by our protocol in the controlled environment.

4.3.1 Parameter selection

We have studied how our system behaves with group sizes ranging from 3 to 10 users. We have made tests for $n = 3$, $n = 4$, $n = 5$ and $n = 10$. Then, we have interpolated the values for groups of 6, 7, 8 and 9 users. The reason for this selection is the following: Google answers around 100 million of queries every day which represents an average value of 1157 queries per second. Those queries can be modeled using a Poisson distribution. Then, if we have an average value of 11.57 queries per hundredth of a second, the probability of generating a group of $n = 3$ queries is close to 1 (see Figure 2 for different values of n). According to those results, every hundredth of a second, the *central node* is able to group 3 users willing to submit a query to the web search engine.

Regarding the length of the keys (parameter l), at the present time $l = 1024$ bits is considered computationally safe [19]. According to that, we have tested our scheme with a smaller length ($l = 768$ bits) and with a larger one ($l = 1296$ bits). In this way, we can examine how the key length influences the system's performance.

We used the Google web search engine in our tests. Each client had a list of 1000 different queries which were selected randomly. The results presented

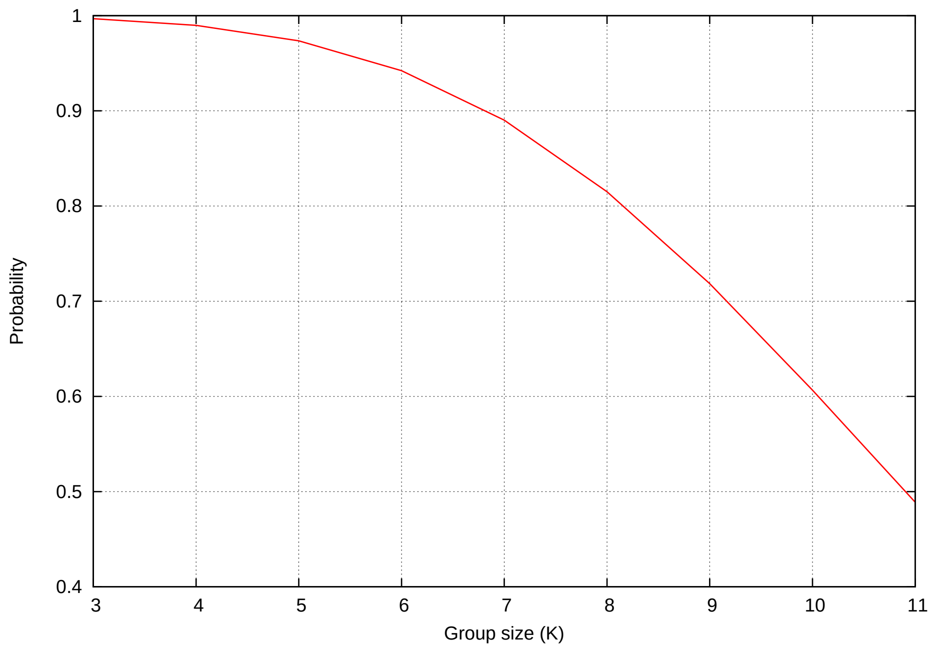

Fig. 2. Probability of forming a group of n users in one hundredth of a second

in this paper are the average of all the values obtained by each client when submitting 1000 queries to Google.

4.3.2 Detailed time analysis

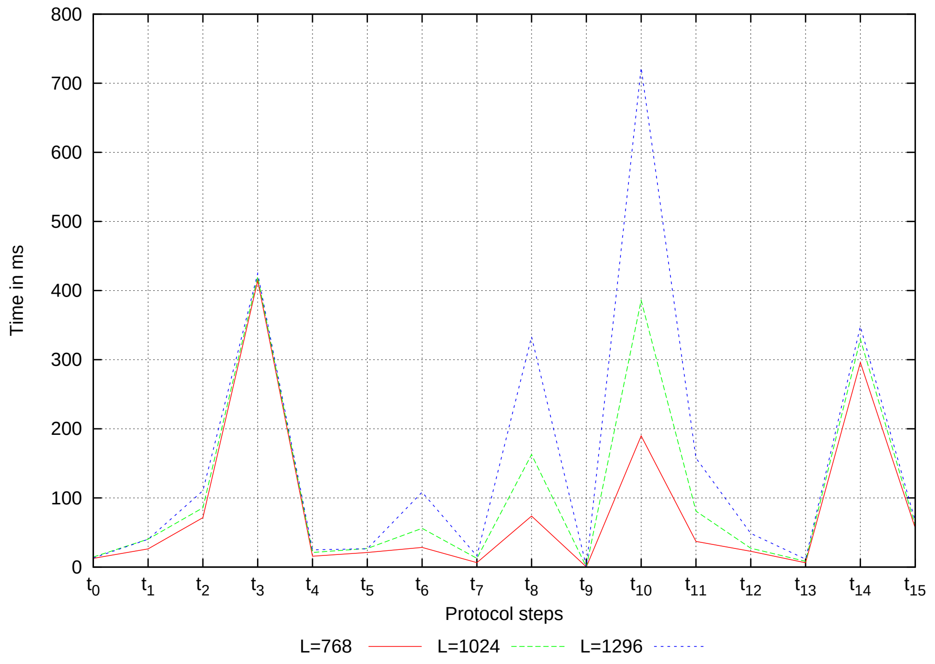

Fig. 3. Partial times with a group of three users ($n = 3$) using different key lengths

Figure 3 shows the partial times for $n = 3$ and for each key length l. Note that, as expected, network times (T_N) are approximately the same for all the three key lengths. Similar results have been obtained for $n = 4$, $n = 5$ and $n = 10$ in the sense that network times are almost the same within the same n value regardless of the key length l. On the other hand, the time interval for cryptographic operations T_C is proportional to l. The most costly operations are the re-masking/permuting process (t_8) and the encryption of the ciphertexts (t_6). Nevertheless, these time intervals are significantly lower than the total cost of the network operations (T_N).

Figure 4 shows how the time intervals are affected by the parameter n once the key length l has been fixed. Note that in this case, the time intervals devoted to cryptographic operations are similar for different values of n. The exception occurs in t_8 (the re-masking/permuting step), where the number of ciphertext involved in the computations are exactly n. Network times vary moderately when n grows.

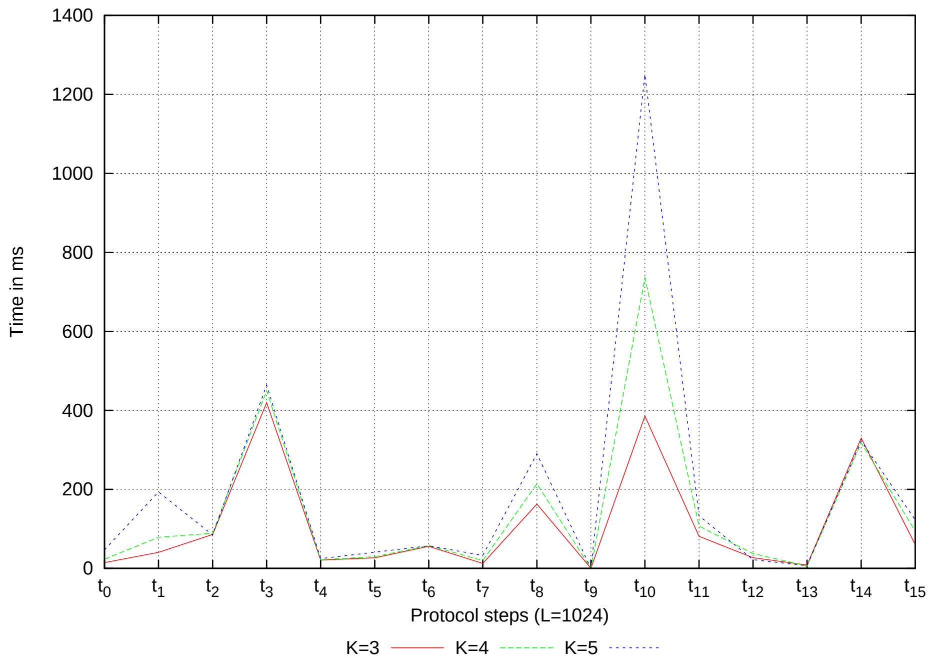

Fig. 4. Partial times with a fixed key length of $l = 1024$ bits and $n = 3, 4, 5$

Both figures show that t_{10} is the protocol's step that introduces the highest overhead into the process. This is the time interval required to receive the results from the re-masking/swapping step. It grows when n or l increase. The cause of that is the sequential order of the re-masking and permuting operation. In this step, each user must wait until the last user has broadcast the final round.

Table 3
Average time (in milliseconds) required to submit a query using our proposal

Group	Key size		
Size	768	1024	1296
3	1451.27	1901.44	2632.52
4	1912.20	2478.94	3603.97
5	2317.97	3311.29	4813.77
10	4896.42	8898.16	14325.73

4.3.3 Global time cost results

Table 3 shows the average time (in milliseconds) required to submit one query using our proposal in the controlled environment. Results are given as a function of the key length l and the group size n.

These values include the time to submit a query to Google, which is around 300 ms (see t_{14} in Figure 3). In scenarios with three users ($n = 3$), there is an increase of 1200 to 2300 milliseconds. The final value depends on the key length.

Figure 5 shows, for each key length l, how the time interval increases when the group size n increases. Thus, the time required to use the presented system grows linearly with the number of users.

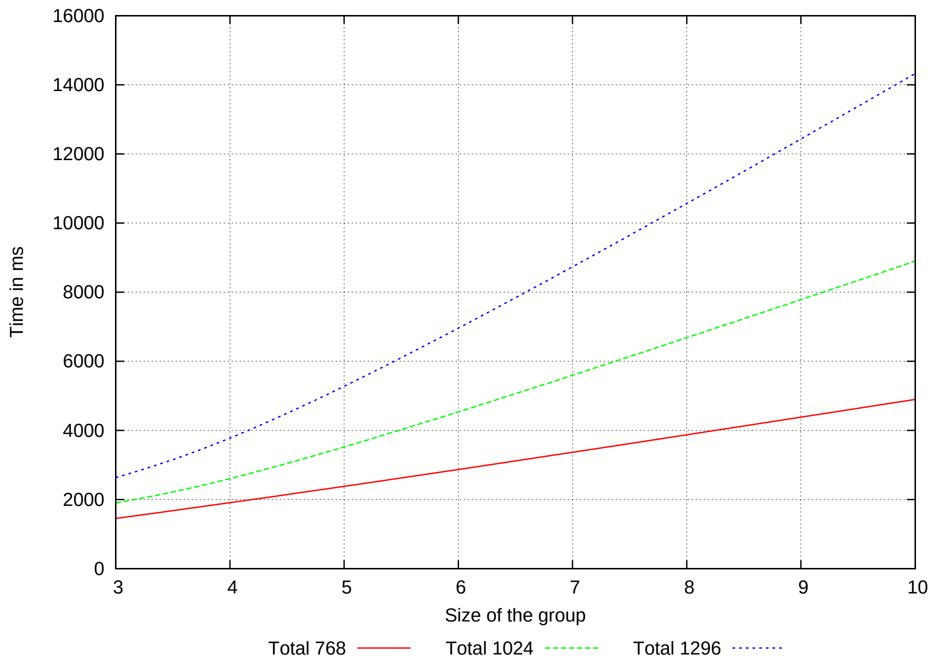

Fig. 5. Average time required to submit a query with our system

Figure 6 shows that the time costs are higher when the number of users and the key length grow. When using a small key length (768 bits), it is not advisable to form groups larger than seven users. If a key of 1024 bits is in use, groups should not be larger than 3 or 4 users. For keys of 1296 bits, the best configuration is a maximum set of 3 users.

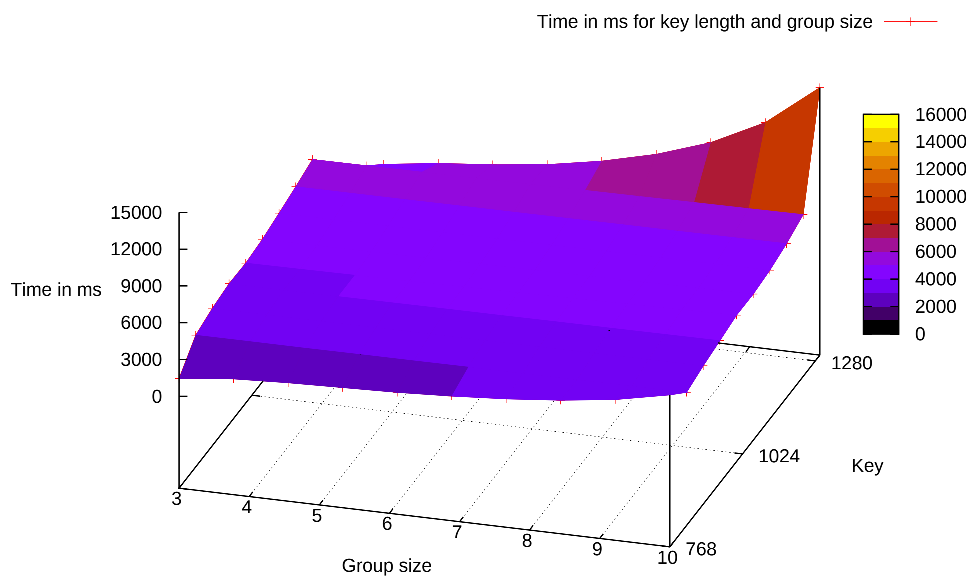

Fig. 6. Cost in time based on the number of users and the length of the keys

4.4 Open environment

The results obtained in the controlled environment are not realistic. However, they are useful to understand the effects of the group size and the key length on the system.

Section 4.4.1 presents the parameters used in the open environment. Finally, Section 4.4.2 shows the time intervals obtained in the open environment.

4.4.1 Parameter selection

According to the time obtained in the controlled environment (see Section 4.3.3), we used $n = 3$ and $l = 1024$ in the open environment.

On one hand a key length below 1024 is not considered safe [19], on the other hand, a key length larger than 1024 increases the computing time. Thus, a length of 1024 bits was selected as the best trade-off between both concepts.

Regarding the group size, we performed our tests with $n = 3$ because in the controlled environment the query delay was larger than 2.5 seconds for bigger groups (see Table 3).

4.4.2 Time analysis

Figure 7 shows the time intervals with $n = 3$ and $l = 1024$ in the controlled and open environments. As expected, the cost of the cryptographic operations is very similar in both environments. Only t_8 is slightly bigger because some computers in the open environment have a slower CPU than the computers in the controlled environment.

The network time is notably higher in the open environment. There, the time interval required to connect with the central node t_1 increases by 700 ms. Besides, it can be noticed a slightly increment in $\{t_3, t_4, t_7, t_9, t_{12}, t_{14}\}$. In t_{10}, the users do the operations sequentially. Thus, each operation only experiences a slightly increment but the aggregation of all these increments results in a significant delay in comparison with the controlled environment. Finally, the most significant overhead is introduced by t_{15} (in this step, the answers from the web search engine are broadcast to the group members). This overhead should be evaluated and improved in a future version of the proposed protocol.

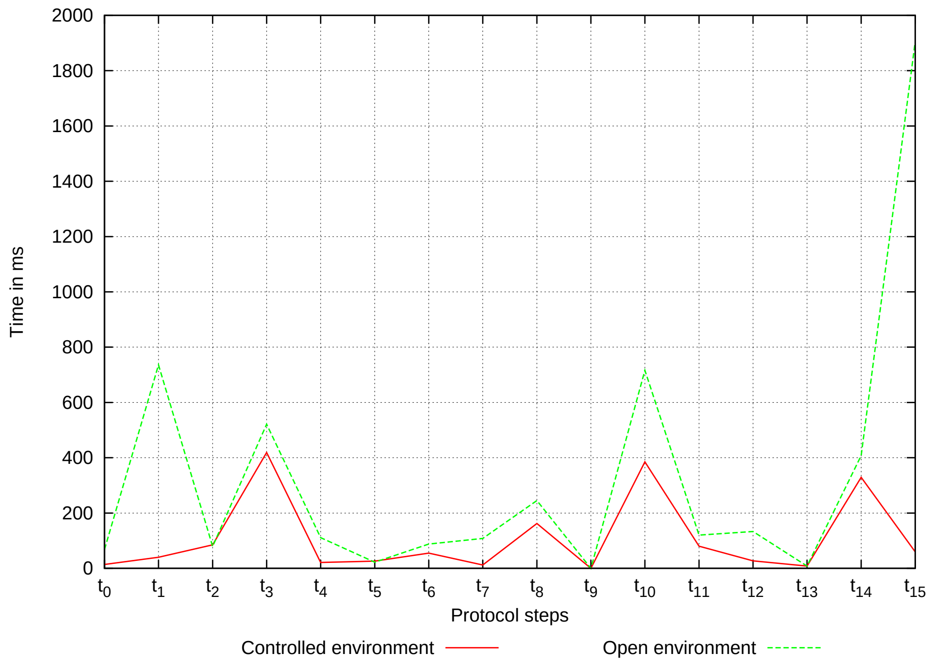

Fig. 7. Partial times with a key length of $l = 1024$ bits and $n = 3$ in the controlled and open environments

As a result of our tests in the open environment, we got a total delay of 5.2 seconds using the UUP protocol with groups of three users ($n = 3$) and a key length of 1024 bits. According to that, our scheme reduces significantly the query delay achieved by previous proposals [10].

4.5 Operational considerations

In previous sections, we have discussed the technical details of the proposed system regarding implementation issues and equipment requirements. Besides, we have also analyzed the computation and communication costs. However, there are other operational considerations that should be taken into account when deploying this scheme in a real scenario.

First of all, from the strictly technical point of view, the proposed scheme provides a certain level of privacy in front of the web search engine at the cost of introducing an affordable delay in the query process. However, from the user point of view, another cost is added in the sense that users are submitting queries from other users. Obviously, a certain user may not be comfortable with the content of other users' queries and may not desire to submit them to the web search engine. This problem can be solved using a filter that discards queries depending on their content. This filter must be carefully designed in order to not require the final user to perform an excessive amount of work. How to implement this is outside the scope of this paper and requires further research. Two possible directions to address this issue would be: (i) allow the users to select categories of words to be discarded. These categories would be based on the ones defined by the Open Directory Project (ODP) [20]; and (ii) use ontologies to process the content of the queries. [21,22] are examples of the work done in this field.

On the other hand, the proposed scheme is based on a central server that connects different users who want to submit a query. Note that the workload of the central server is really low since it only offers to users contact details about other users. The rest of the steps needed to form the group are entirely performed by the users. According to that, it is possible to assume that some central nodes can be voluntary working to improve the privacy of the users. In fact, the Tor network, which has a similar privacy preserving objective, is based on the use of voluntary nodes (more than 1800 [23]). Besides, the workload of a Tor node is higher than the workload of the central node used in our scheme.

Finally, another operational consideration arises regarding the number of users needed to form a group and the time each user has to wait until that number is available. The main objective of our proposal is not to offer anonymity to the users who submit queries to a web search engine. Instead of that, it provides non-accurate user profiles to the web search engine. According to that, the system is still valid if the users submit some of their own queries. This can be implemented in the client application using a timeout. If, after this timeout, the group has not been formed, the user can send her query to the web search engine directly.

The correctness of this measure can be controlled by keeping track of the total number of queries that have been submitted directly. This number must be kept below a predefined threshold in order to provide a useless profile to the web search engine. Defining the correct value for this threshold requires a detailed study which is not the purpose of this paper. Nevertheless, the work presented in [24] can give us an approximation of this value. In [24], the system submits τ fake queries for each legitimate query. In order to define the privacy level which is achieved, the system uses one vector to store all the fake queries that have been submitted and another vector to store the legitimate ones. The similarity between both vectors defines the privacy level achieved. The results provided by [24] show a fairly good privacy level when $2 \leq \tau \leq 4$.

5 Conclusions and further research

Web search engines are gathering more and more private information from individuals since they have become the main tool for finding information. In the last years, specific proposals for protecting the privacy of the users of web search engines have been presented. Those proposals improve the general schemes used so far, like anonymous web browsing, but they still experience a moderate delay in the query response.

In this paper, a novel protocol for protecting the users' privacy when dealing with a web search engine has been proposed. Our scheme does not require any change in the server side and, moreover, the server is not required to collaborate with the user. The proposed protocol has been implemented to prove its functionality. Statistical results of the protocol's performance show that the presented scheme improves previous proposals. In addition to that, these results also prove that this protocol can be applied in real scenarios.

Our future research will focus on two different lines:

- We will try to improve the performance of the protocol by reducing the delay introduced. On one hand, we should review the re-masking/swapping process. At the present implementation, the sequential order followed by each user in this step increases the total cost. This procedure is derived from the cryptographic primitives used. Therefore, other approaches should be analyzed in order to improve this protocol step. On the other hand, the time required to distribute the received answers to the other users should also be reduced.

- Our security analysis assumes that the users follow the protocol correctly and no collusion between entities is allowed. Relaxing those assumptions implies that more strong cryptographic operations should be included in the protocol to ensure its security. This has to be studied carefully because the use of expensive cryptographic mechanisms may introduce an unaffordable query delay.

Disclaimer and acknowledgments

The authors thank the anonymous reviewers for their valuable comments that helped to improve the quality of this paper. The authors are with the UNESCO Chair in Data Privacy, but they are solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. Thanks go to Alexandre Garcia for his help in implementing the protocol. This work was partly supported by the Spanish Ministry of Education through projects TSI2007-65406-C03 "E-AEGIS" and CONSOLIDER CSD2007-00004 "ARES", and by the Government of Catalonia under grant 2005 SGR 00446.

References

[1] D. Fallows, "Search Engine Users: Internet searchers are confident, satisfied and trusting, but they are also unaware and naïve", *Pew/Internet & American Life Project*, 2005.

[2] D. Sullivan, "comScore Media Metrix Search Engine Ratings", *comScore*, `http://searchenginewatch.com`, 2006.

[3] Google History, `http://www.google.com/history`, 2009.

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

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

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

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

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

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

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

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

[12] M. Barbaro and T. Zeller, “A face is exposed for AOL searcher No 4417749”, *New York Times*, August 2006.

[13] Y. Xu, B. Zhang, Z. Chen, and K. Wang, “Privacy-Enhancing Personalized Web Search”, *International World Wide Web Conference*, pp. 591–600, 2007.

[14] Y. Desmedt and Y. Frankel, “Threshold cryptosystems”, *Advances in Cryptology – CRYPTO’89*, *Lecture Notes in Computer Science*, vol. 335, pp. 307–315, 1990.

[15] 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.

[16] D. Chaum and T. Pedersen, “Wallet databases with observers”, *Advances in Cryptology – CRYPTO’92*, *Lecture Notes in Computer Science*, vol. 740, pp. 89–105, 1992.

[17] M. Abe, “Mix-networks on permutation networks”, *Advances in Cryptology - Asiacrypt’99*, *Lecture Notes in Computer Science*, vol. 1716, pp. 258-273, 1999.

[18] Sun Microsystems, JAVA Programming language, `http://java.sun.com`, 2008.

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

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

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

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

[23] Tor Node Status, `https://torstat.xenobite.eu/`, 2009.

[24] T. Kuflik, B. Shapira, Y. Elovici, A. Maschiach, “Privacy Preservation Improvement by Learning Optimal Profile Generation Rate”, *Lecture Notes in Computer Science*, vol. 2702, pp. 168–177, 2003.