Multi-party Private Web Search with Untrusted Partners

Cristina Romero-Tris, Jordi Castellà-Roca, and Alexandre Viejo

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

Abstract. Web search engines are tools employed to find specific information in the Internet. However, they also represent a threat for the privacy of their users. This happens because the web search engines store and analyze the personal information that the users reveal in their queries. In order to avoid this privacy threat, it is necessary to provide mechanisms that protect the users of these tools.

In this paper, we propose a multi-party protocol that protects the privacy of the user not only in front of the web search engine, but also in front of dishonest internal users. Our scheme outperforms similar proposals in terms of computation and communication.

Keywords: privacy, web search engines, private information retrieval.

1 Introduction

Search on the Internet is a frequent activity for many users throughout the world. Web search engines (WSEs) are tools which allow information retrieval from this huge repository of data. There are many WSEs in the market, such as Google, Bing, Yahoo, etc.

When a user wants to search a term in a WSE, she types the keywords and submits her query. Then, the WSE applies information retrieval techniques to select and rank the results. After that, the user evaluates the list of pages and gets the information.

Along with this process, the WSE builds a profile of this user based on her queries. For example, in its Privacy Center [1], Google states that its servers automatically record requests made by users. These "server logs" include 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.

Google uses cookies for several purposes such as identifying the users in order to improve their search results and track their trends, and also storing their preferences. Google also uses the cookies in its advertising services to help companies serve and manage the promotion of their products across the web. This is called AdSense and it represents a large source of income for Google.

M. Rajarajan et al. (Eds.): SecureComm 2011, LNICST 96, pp. 261–280, 2012.

Besides the financial gain for WSEs, profiling is a threat for the privacy of the user. The different logs stored by a WSE contain sensitive data that can be combined to disclose information of a certain individual. In order to do that, it is necessary to find the identity of the user. One way of doing it is to use the IP address and the cookies stored in the logs. Thus, queries that come from the same IP address or from a browser with a certain cookie are used to build the same profile. Note that users cannot rely on deleting the cookies and on the use of different IP addresses. The renewal policy of dynamic IP addresses depends on the network operator. Furthermore, some users might require static IP addresses.

In addition to IP addresses and cookies, people can reveal their personal identity in their queries. In fact, [2] indicates that 94.82% of users have searched their own name at least once. Moreover, many other queries such as the place where they live, their job, or even the car they drive, can also be a method of tracing users and revealing their identity.

Once a user is identified, the WSE can link her identity with the queries she made. According to [2], around 85% of users have searched for information that they would not want their parents or their employers to know about. For example, queries about health, sexual orientation, politics, religion, etc. can be considered extremely sensitive information for the owners. Hence, the queries of a user should be protected and never revealed to third parties.

Some incidents in the past have shown that WSEs are not capable of protecting the privacy of the users. For example, in 2006 AOL released a file with twenty million searches generated by its users [3]. This incident had serious consequences since personally identifiable information was present in many of the queries. Another example of privacy risks with WSEs is the subpoena that Google suffered in 2006 [4]. On that occasion, the Justice Department of U.S.A. tried to compel Google to provide millions of Internet search records.

Such events indicate that users should not trust the companies behind the WSEs. Therefore, it is necessary to propose alternatives that prevent the WSEs from knowing the sensitive information of the users.

2 Previous Work

The problem of private web search has been widely discussed in previous literature. In this section, the main contributions to this subject are described.

The problem introduced in this paper is similar to the Private Information Retrieval (PIR) problem [5]. However, PIR protocols are not suitable for WSEs because they assume that the server which holds the database collaborates with the user. In the WSE scenario this assumption cannot be made because WSEs have no motivation to protect the privacy of the users since it would limit its profiling objectives.

Another solution to maintain the privacy of the users is to use a proxy. There are several companies (*e.g.* Scroogle [6], anonymizer.com [7]) that offer a service in which the clients can redirect the traffic to their servers. As a result, requests seem to be originated by these servers and have no reference to the IP address of the client. Nevertheless, this is not the best solution to protect the privacy of the users because profiling could be done at the proxy, hence, instead of trusting the WSE, users have to trust the proxy.

Onion routing is a technique to establish anonymous channels that preserve the privacy of the users. An example of this is the Tor anonymity network which is described in [8]. The authors in [9] propose to use the anonymous channels to submit queries to the WSE. The main drawback of this scheme is that the encryption and decryption process at the onion routers make the search process too slow. According to [9], the cost of submitting one query to Google is about 10 seconds on average. This means that users would spend 25 times longer doing each query. This query delay is very high for a tool that is expected to be used quite frequently.

Another alternative is the use of a query obfuscation protocol such as GooPIR [10] or TrackMeNot [11]. These protocols generate a stream of automated queries where the real queries are blended into. As a result, the WSE is not able to create a correct profile. GooPIR uses a Thesaurus to obtain the words which are mixed with the real queries. Consequently, the fake queries are single words, while full sentences are not addressed. TrackMeNot is a plugin for Mozilla Firefox that generates dynamic queries using RSS feedback. These queries can be words or sentences, and they are periodically submitted to the WSE.

These obfuscation protocols have a major disadvantage: machine-generated queries do not have the same features as the human-generated queries. The works presented in [12] and [13] argue that it is possible to distinguish real queries from automated queries. For example, [13] develops a classifier which is very accurate in identifying TrackMeNot queries, with a mean of misclassification around 0.02%.

Another approach is to use a multi-party protocol, which is not affected by the misclassification issue of the single-party ones. Besides, they are generally faster than the schemes based on anonymous channels. In this kind of protocols, a group of users is created. Then, a user asks another component of the group to submit her query and send back the result.

In [14], the authors propose a multi-party protocol named Useless User Profile (UUP). The basic idea beneath this system is that a central node puts users into dynamic groups where they securely exchange their queries. As a result, each user submits a partner's query and not her own and, hence, she obtains a distorted profile. This protocol achieves a query delay of 5.2 seconds. This time significantly outperforms previous proposals. However, the UUP protocol has a major disadvantage. It is not secure in presence of malicious internal users. This means that a dishonest user can learn the queries of the rest of members of the group.

The authors of [15] use an scenario which is similar to the one proposed in [14]. However, they argue that the level of security of [14] is not sufficient. Hence, they modify the UUP protocol in order to be resilient against some attacks. Nevertheless, the drawback of their proposal is that it uses expensive cryptographic tools (*i.e.* double encryptions) that introduce an unaffordable query delay. In fact, the authors remark that executing their protocol is twice as expensive as in [14].

Finally, another similar approach is presented in [16] and [17]. The idea of both proposals is to minimize the role that the central node plays in the protocol. On one hand, [16] proposes to use a preestablished network with the topology of a complete graph. On the other hand, [17] proposes to employ already developed social networks (*e.g.* Facebook).

The main drawback of both proposals is that the groups are static (same members in every execution of the protocol). This means that their protocols are more vulnerable in front of an internal attack (*e.g.* the attacks proposed in [15]).

2.1 Contribution and Plan of This Paper

In this paper, we present a new 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 [14], and requires less computation and communication than [15].

Section 3 introduces the background and tools that the protocol uses. Section 4 describes the scenario and the privacy requirements. The protocol is detailed on Section 5. Section 6 and 7 analyze its privacy and performance respectively. Finally, Section 8 concludes the paper and reports some future work.

3 Background and Notation

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 (see [18] for more details), n users have a distributed public key y and the corresponding secret key α is divided into n shares α_i, where no single party knows the entire secret. 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.

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

Message Encryption. Message encryption can be performed using the standard ElGamal encryption function [19]. Given a message m and a public key y, a random value r is generated and the ciphertext is computed as follows:

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

Message Decryption. Given a message encrypted with the 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 [20].

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 [21]:

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

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.

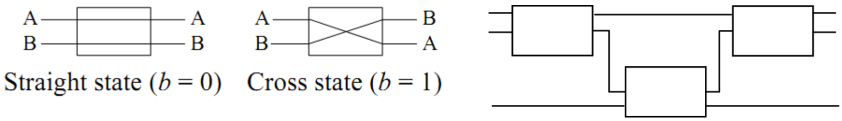
3.3 Optimized Arbitrary Size (OAS) Benes

A Benes permutation network (PN) [22] 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 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 [23], it is necessary to introduce a 3 x 3 network like Figure 1(b) shows. Using 2 x 2 switches and 3 x 3 networks recursively it is possible to construct a network of any size.

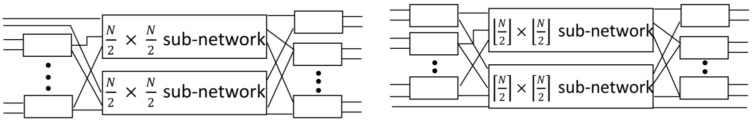
Optimized Arbitrary Size (OAS) Benes [24] is an extension of AS Benes that reduces the number of necessary switches in the network. The way of constructing the OAS-Benes depends on the parameter N:



(a) States of a 2 x 2 switch (b) 3 x 3 network

Fig. 1. Basic elements of an OAS-Benes

- 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 2(a) 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 2(b) illustrates this construction.


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

(a) for even N (b) for odd N

Fig. 2. Construction of OAS-Benes

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:

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

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

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 is such a way that no user knows the overall permutation between the inputs and the outputs.

According to [24], 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. Since the construction of the OAS-Benes is mechanical, the users can build it without any cooperation between them or from another entity.

In order to obtain a secure permutation, the condition that must be satisfied is that the honest users control, at least, $S(N)$ switches. We denote as λ the minimum number of honest users that the system requires. For example, consider 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 [24], the $\lambda = 3$ honest users must control 17 or more switches. This means that every user must control $\left\lceil \frac{17}{3} \right\rceil = 6$ switches. Therefore, the scheme needs at least (6 *switches per user* $\times$ 6 *users*) $= 36$ switches that will be fairly divided among the n users. Consequently, the system requires $t = \left\lceil \frac{36}{17} \right\rceil = 3$ OAS-Benes $PN^{(8)}$.

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

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

3.4 Plaintext Equivalence Proof (PEP)

PEP [25] is an honest-verifier zero-knowledge proof protocol based on a variant of the Schnorr signature algorithm [26]. 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:

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

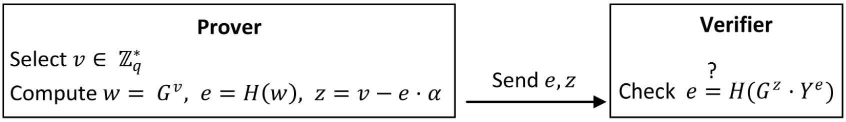

Prover
Select $v \in \mathbb{Z}_q^*$
Compute $w = G^v$, $e = H(w)$, $z = v - e \cdot \alpha$
Send e, z
Verifier
Check $e \overset{?}{=} H(G^z \cdot Y^e)$

Fig. 3. PEP protocol

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.

3.5 Disjunctive PEP (DISPEP)

DISPEP [25] is an extension of the PEP protocol. In this case, a user proves 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 4. Without loss of generality, in Figure 4, we assume that the prover is showing $m_a = m$.

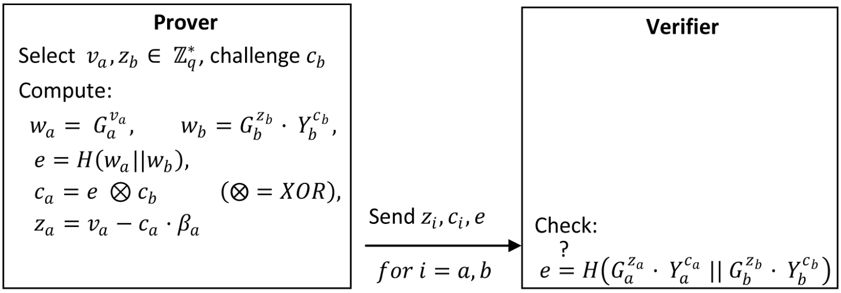

Prover
Select $v_a, z_b \in \mathbb{Z}_q^*$, challenge c_b
Compute:
$w_a = G_a^{v_a}$, $w_b = G_b^{z_b} \cdot Y_b^{c_b}$,
$e = H(w_a || w_b)$,
$c_a = e \otimes c_b$ $(\otimes = XOR)$,
$z_a = v_a - c_a \cdot \beta_a$
Send z_i, c_i, e
$for\ i = a, b$
Verifier
Check:
$e \overset{?}{=} H(G_a^{z_a} \cdot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

Fig. 4. DISPEP protocol

4 System Model

4.1 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 is the entity that 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.
- *Web search engine.* It is the server that holds the database. As previously mentioned, WSEs have no motivation to protect the privacy of their users.

4.2 Protocol Overview

The idea of the protocol is to create a group of users who collaborate in order to make searchs in 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. As a result, the WSE cannot create a reliable profile of any particular individual.

The protocol requires that neither the WSE nor the users of the group learn which query belongs to each user. In order to do this, the users execute a multiparty protocol that works as follows: a central node creates a group of n users. Then, the required OAS-Benes networks are 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. 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 broadcast the result.

4.3 Privacy 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 is not able to construct a reliable profile of any user.

5 Protocol Description

The protocol is composed by four phases that the users execute sequentially.

5.1 Group Setup

Every user who wants to submit a query to the WSE, contacts the central node. When the central node has received n requests, it creates a group $\{U_1, \ldots, U_n\}$. Then, the n users are notified 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.

5.2 Permutation Network Distribution

As stated in section 3.3, 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, is later detailed in Section 7.1.

Knowing the parameters n, N, and λ, the users calculate the value of t using the formula defined on Section 3.3. 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 in 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, 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 illustrates the operation of this function.

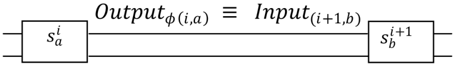


Fig. 5. Correlation between the outputs of a switch and the inputs of the next

5.3 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) Broadcasts y_i to the other members of the group.
   (e) Checks that $h_j = \mathcal{H}\,(y_j)$ for $j = (1, \ldots, n)$.
   (f) Calculates the group key using the received shares:
   $y = \prod_{1 \leq j \leq n} y_j = g^{a_1} \cdot g^{a_2} \cdot \ldots \cdot g^{a_n}$

5.4 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:
$$E_y(m_i, r_i) = (c1_i, c2_i) = c_i^0$$
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 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:
$$c_{i-1}^{2l-1} = E_y(m_1, r_1), \quad c_{i-1}^{2l} = E_y(m_2, r_2)$$
$$e_{i-1}^{\pi(2l-1)} = E_y(m_1', r_1'), \quad e_{i-1}^{\pi(2l)} = E_y(m_2', r_2')$$
   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 the two following 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 $\{c_1, \ldots, c_n\}$ 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, 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:
$$m^i = \frac{c2_i}{c1_i^{\alpha_i}(\prod_{j \neq i} c1_i^{\alpha_j})}$$

6 Privacy Analysis

This section analyzes the behaviour of the protocol regarding the privacy requirements that appear on Section 4.3. Basically, these requirements demand that, at the end of the protocol, no query can be linked to the user who generated it.

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.

6.1 Dishonest User

The ElGamal cryptosystem is sematically secure under the Decisional Diffie-Hellman assumption. 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 random 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 of the protocol, 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.* In this case, the attacker can decrypt the queries at any step of the protocol.
2. *The dishonest user ignores the key but knows the overall permutation.* In this case, 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 on Section 3.1. 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 $y_i = g^{a_i}$, then she broadcasts a commitment to that share using a cryptographic function $\mathcal{H}(y_i)$, and then she 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 remains random and private. The requirement that must be satisfied is that there must be at least one permutation network controlled by honest users. This means that the proposed scheme needs a quantity of PNs that 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 [24], 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.

6.2 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.

However, 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. In a similar situation, an attacker could send many requests to the central node such that it is likely that she controls a large fraction of the group.

In order to prevent these attacks, the authors of [15] propose a solution that can be straightforwardly applied to our protocol. Their solution consists in a joint coin tossing scheme that uniformly distributes the parties controlled by the central node among all the groups executing the protocol. However, their proposal has two obstacles that may affect its practical deployment:

1. The number of parties controlled by the central node must be small in comparison with the number of users ready to execute the protocol at a certain time. In [15], the authors consider the case of millions of users running the protocol, while the dishonest central node only controls a few thousands of them.
2. Executing the joint coin tossing scheme is expensive. Therefore, [15] proposes to reuse the groups in several consecutive executions of the protocol. However, the users of the same group may not want to submit another query

at the same time. On the other hand, sharing a group several times with the same user increases the probability of learning one of her queries.

6.3 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. The WSE receives the queries from all the members of the group and returns the results.

The WSE can link each query with the user who submitted it and include that information on her profile. Since a user U_i does not submit her own query but the query of another participant, her profile is distorted. Hence, after several executions of the protocol, the profile of U_i that the WSE owns is useless.

7 Performance Analysis

The objective of this section is to analyze the performance of our proposal and to compare the results with other similar proposals. Our proposal is compared with two similar approaches: the scheme proposed by [14] and the scheme presented in [15]. Since the work presented in [15] does not include simulations nor a query delay estimation in a real environment, we decided to analyze the protocols theoretically. For this purpose, we analyze the protocol regarding the required computation time and the number of messages that need to be exchanged.

7.1 Parameter Selection

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).

Size of the Group and Key Length. 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 bigger size of the group, the more privacy the 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 quick. According to [14], 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 [14] conclude that $n = 3$ is the most realistic group size. As stated in [14], the probability of forming a group of $n = 3$ users in a hundredth of a second is close to 1.

For this reason, in the subsequent performance analysis we present the results obtained for $n = 3$ users. For a more complete comparison, we also show the results for a group size of $n = 4$ and $n = 5$ users.

Regarding the key length, according to [14] and [27], a 1024-bit key length is considered computationally safe. In addition, the work presented in [28] 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.

Minimum Number of OAS-Benes PNs. The minimum number of OAS-Benes PNs, denoted as (t), is calculated according to the formula defined on Section 3.3. 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 analyis.

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 wihout 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 6.2, in a scenario with a single honest user and $n - 1$ dihonest 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.

7.2 Analysis of the Computation Time

Next, we analyze the computation time needed in the execution of [14], [15] and our proposal. More specifically, we focus on the amount of modular exponentiations that every user must perform in each execution of the protocol.

There are some parts of the protocol of [15] 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 [14] 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 1 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.

Table 1. Modular exponentiations average time for one user

Castellà et al. [14]	$(3n+3)\cdot\tau_{1024}$
Lindell et al. [15]	$6n\cdot\tau_{1024}+5n\cdot\tau_{2048}-\tau_{1024}+2\cdot\tau_{2048}$
Our Proposal	$\left(n+3+\frac{25\cdot t\cdot S(n)}{n}\right)\cdot\tau_{1024}$

Figure 6 shows the calculated times for a group size of 3, 4 and 5 users. The results indicate that [14] obtains the lowest computation time. This happens because [14] does not use any mechanism to protect the participants against dishonest users. Since [15] 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 [15]. For example, for $n = 3$ users, our proposal requires approximately one second more of computation time than [14], while [15] needs 3 more seconds than [14].

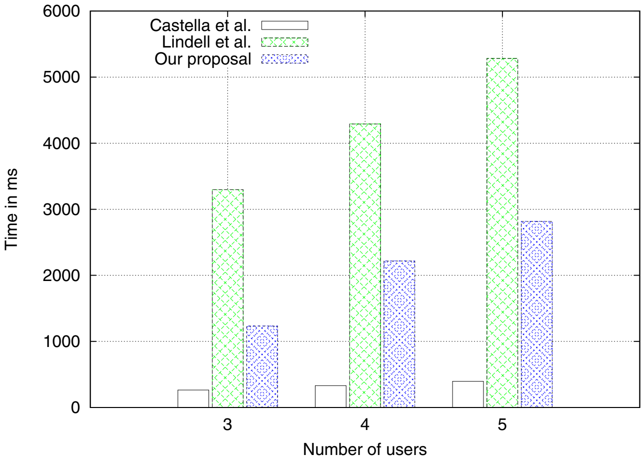


Fig. 6. Comparison of modular exponentiations times per user

7.3 Analysis of the Number of Messages

In order to analyze the performance of the protocol, another relevant parameter is the usage of the network. Table 7.3 reflects the number of messages that every user sends in each execution of the protocol.

Table 2. Average number of messages sent by each user

Castellà et al. [14]	$3n - 1 - \frac{2}{n}$
Lindell et al. [15]	$4n - 2 - \frac{2}{n}$
Our Proposal	$4n - 4$

Figure 7 represents the number of messages sent when 3, 4 or 5 users jointly execute the protocol. Although the number of messages is similar in the three proposals, the results indicate that the number of messages sent in [14] is lower than in [15] and in our proposal. The results also indicate that our proposal requires less message deliveries than [15].

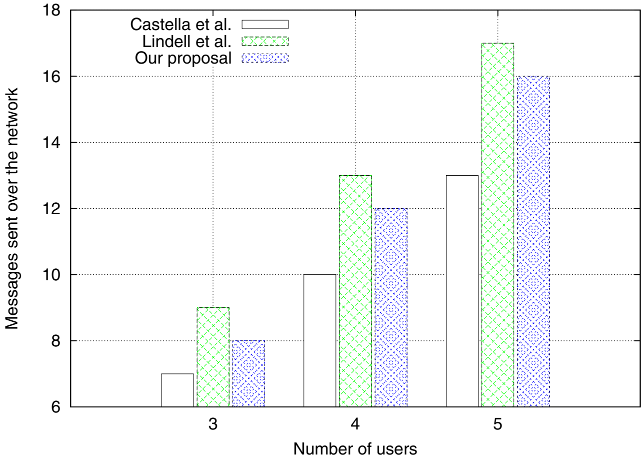


Fig. 7. Comparison of messages sent in each protocol per user

7.4 Additional Remarks

There is another difference between the protocol of [15] and our proposal that affects the performance. In [15], in order to detect a dishonest user, the participants must wait until the last phase of the protocol (i.e., when the last user broadcasts the results). At this point, if they detect any irregularity, the honest users discard the obtained results and begin a new execution with a new group.

On the other hand, our proposal is able to detect a misbehaviour earlier. For example, if the first user is dishonest, her misbehaviour can be detected immediately after she sends her zero-knowledge proofs. After the detection, the rest of users logout and start a new execution.

In conclusion, in the presence of one or more dishonest users, the users waste more time running the protocol of [15] than if they execute our proposal.

8 Conclusions and Future Work

Users frequently reveal personal information in the queries that they submit to WSEs. WSEs store this information and use it to improve the search results and for targeted advertising. In order to avoid this situation, this paper proposes a protocol that protects the privacy of the users from web search profiling.

The proposed protocol has been analyzed in terms of privacy and performance. The privacy analysis shows that the users are protected in front of the WSE and of dishonest internal users. Regarding the performance, the protocol ouperforms similar proposals with the same level of privacy.

The future work will focus on two different lines. The first line is the implementation of the proposed protocol and deployment in a real scenario. Making simulations in this scenario will allow to estimate the real query delay and compare its performance results with similar proposals. The second line of future work will focus on the search of a peer-to-peer solution that does not require the use of a central node in order to create the groups.

Disclaimer and Acknowledgments

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. This work was partly supported by the Spanish Ministry of Science and Innovation through projects TSI2007-65406-C03-01 "E-AEGIS", CONSOLIDER CSD2007-00004 "ARES" and PT-430000-2010-31 "Audit Transparency Voting Process", and by the Government of Catalonia under grant 2009 SGR 1135.

References

1. Google Privacy Center (2011), http://www.google.com/privacy
2. Conti, G., Sobiesk, E.: An honest man has nothing to fear: user perceptions on web-based information disclosure. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, pp. 112–121 (2007)

3. Barbaro, M., Zeller, T.: A Face is Exposed for AOL Searcher No. 4417749. New York Times (August 2006)
4. Hafner, K., Richtel, M.: Google Resists U.S. Subpoena of Search Data. New York Times (January 2006)
5. Ostrovsky, R., Skeith III, W.E.: A Survey of Single-Database Private Information Retrieval: Techniques and Applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)
6. Scroogle (2011), `http://scroogle.org`
7. Anonymizer (2011), `http://www.anonymizer.com`
8. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium, p. 21 (2004)
9. Saint-Jean, F., Johnson, A., Boneh, D., Feigenbaum, J.: Private Web Search. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society – WPES 2007, pp. 84–90 (2007)
10. Domingo-Ferrer, J., Solanas, A., Castellà-Roca, J.: h(k)-private information retrieval from privacy-uncooperative queryable databases. Journal of Online Information Review 33(4), 720–744 (2009)
11. TrackMeNot (2011), `http://mrl.nyu.edu/dhowe/trackmenot`
12. Chow, R., Golle, P.: Faking contextual data for fun, profit, and privacy. In: Proceedings of the 8th ACM Workshop on Privacy in the Electronic Society – WPES 2009, pp. 105–108 (2009)
13. Peddinti, S.T., Saxena, N.: On the Privacy of Web Search Based on Query Obfuscation: A Case Study of TrackMeNot. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 19–37. Springer, Heidelberg (2010)
14. Castellà-Roca, J., Viejo, A., Herrera-Joancomartí, J.: Preserving user's privacy in web search engines. Computer Communications 32(13-14), 1541–1551 (2009)
15. Lindell, Y., Waisbard, E.: Private Web Search with Malicious Adversaries. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 220–235. Springer, Heidelberg (2010)
16. Reiter, M., Rubin, A.: Crowds: anonymity for Web transactions. ACM Transactions on Information and System Security 1(1), 66–92 (1998)
17. Viejo, A., Castellà-Roca, J.: Using social networks to distort users' profiles generated by web search engines. Computer Networks 54(9), 1343–1357 (2010)
18. Desmedt, Y., Frankel, Y.: Threshold Cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
19. ElGamal, T.: A public-key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31, 469–472 (1985)
20. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
21. Abe, M.: Mix-Networks on Permutation Networks. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 258–273. Springer, Heidelberg (1999)
22. Waksman, A.: A Permutation Network. Journal of the ACM 15(1), 159–163 (1968)
23. Opferman, D., Tsao-Wu, N.: On A class of Rearrangeable Switching Networks. Bell Systems Technical Journal 50(5), 1579–1618 (1971)
24. Soo, W.H., Samsudin, A., Goh, A.: Efficient Mental Card Shuffling via Optimised Arbitrary-Sized Benes Permutation Network. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 446–458. Springer, Heidelberg (2002)

25. Jakobsson, M., Juels, A.: Millimix: mixing in small batches. DIMACS Technical report 99-33 (1999)
26. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. Journal of Cryptology 4, 161–174 (1991)
27. Recommendation for Key Management, Special Publication 800–57 Part 1, NIST (2007)
28. Kamvar, M., Baluja, S.: 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)