Improving Query Delay in Private Web Search

Cristina Romero-Tris, Alexandre Viejo, Jordi Castellà-Roca
Universitat Rovira i Virgili, UNESCO Chair in Data Privacy
Departament d'Enginyeria Informàtica i Matemàtiques
Av. Països Catalans 26, E-43007 Tarragona, Spain
Email: {cristina.romero, alexandre.viejo, jordi.castella}@urv.cat

***Abstract*—The Internet is a huge repository of information. Web search engines are a basic tool for finding and accessing all this information. However, these tools might also be a threat for the privacy of their users. This happens because users frequently reveal private information in their queries. Web search engines gather this personal data, store it during a large period of time and use it to improve their search results and to increase their economical benefits. In order to avoid this situation, it is necessary to provide web search methods that preserve the privacy of the users. Current proposals in the literature increase significantly the query delay. This is the time that users have to wait in order to obtain the search results for their queries. In this paper, we propose a modification of the Useless User Profile (UUP) protocol. The resulting scheme has been tested in an open environment and the results show that it achieves the lowest query delay which has been reported in the literature. In addition to that, it incentivizes users to follow the protocol in order to protect their privacy.**

I. Introduction

A web search engine (WSE) is a tool designed to search for information on the Internet. There are many examples of WSEs in the market, such as Google, Bing, AOL, etc. In 2007, a survey [1] revealed that 92.44% of respondents chose Google as their most used WSE. Accordingly, this paper focuses on Google's search engine.

Although WSEs play an important role in the use of the Internet, they can also raise concerns regarding the privacy of the users. WSEs are employed by thousand of users every day. This means that a significant amount of their personal information is revealed. WSEs use this information to build profiles that identify the users. These profiles can be employed with different purposes. On one hand, the WSE uses the profile of a user in order to improve her search results, track her trends and store her preferences. On the other hand, her profile may also be used for targeted advertising. This represents a large source of income for the WSE.

A profile is associated with the set of queries that the user submits. Some queries may contain personally identifiable information such as her full name, Social Security number, where she lives, where she works, etc. Some queries may contain sensitive information such as health issues, sexual orientation, politics, religion, etc. When all the queries are put together into the same profile, the privacy of the user can be seriously compromised.

Consequently, the WSEs should never reveal the personal information to third parties. However, a user cannot control how the WSE manages her information. Some incidents in the past years have shown that WSEs cannot be trusted in this matter. In 2006 AOL released a file with twenty million queries generated by its users [2]. In 2006 Google suffered a subpoena where the Justice Department of the U.S.A. required the WSE to provide millions of records with users' queries [3].

These facts alerted some WSE users of the risks of unprotected web searching. In order to protect their queries, the users cannot rely on WSEs. Therefore, they need other alternatives that prevent the WSEs from building their profiles. These alternatives can be classified into two groups: single-party protocols and multi-party protocols. Single-party protocols such as [4] and [5] submit machine-generated queries combined with the queries of the user. Nevertheless, some proposals (e.g. [6], [7]) prove that it is possible to distinguish real queries from automated queries with a very low probability of misclassification (around 0.02%). Another option is to use an anonymous channel such as Tor [8]. However, with this scheme, the sending process is 25 times slower than a direct connection [9]. The main disadvantages of single-party protocols are detailed in [10].

Multi-party protocols are not affected by the misclassification issue of the single-party ones. Besides, they are generally faster than the schemes based on anonymous channels. In a multi-party protocol, a group of users is created. Then, a user asks another component of the group to submit her query and send back the result. However, query delay is still a restriction of this kind of protocols. This is the time that users have to wait in order to obtain the search results for their queries. As stated in [1], most of the users are not determined to sacrifice the speed of the WSE in exchange of their privacy. Hence, when a group of users collaborates in a multy-party protocol, the computation and communication cost must be minimized.

A. Previous work on multi-party protocols

The multi-party protocol presented in [10] is named *Useless User Profile* (UUP). The UUP protocol proposes to put users into dynamic groups where they securely exchange their queries. At the end of the protocol a user submits the query of someone else in the group, not her own. The result is that the WSE cannot create a correct profile. Regarding the efficiency, the protocol in [10] obtains a 5.2 second query delay. This result outperforms significantly other similar proposals.

A similar approach using static groups instead of dynamic ones was proposed in [11] and [12]. In those schemes, the same members participate in every execution of the protocol. Accordingly, the WSE could build a profile from a group

978-0-7695-4531-8/11 $26.00 © 2011 IEEE
DOI 10.1109/3PGCIC.2011.61

but not from a specific user. The main problem with these protocols is that static groups are more vulnerable against an internal attack like the ones which were proposed in [13].

B. Contribution and plan of this paper

In this paper, we propose a modification of the work presented in [10]. On one hand, our system optimizes some steps of the protocol to reduce the delay of each query. On the other hand, these changes incentivize every user to follow the protocol in order to protect their privacy.

The paper is organized as follows. Section II introduces the protocol proposed in [10]. It details the concepts of their system that the new protocol employs later. Section III explains the new protocol. In Section V a privacy analysis is performed. The implementation and simulation of the protocol is reported in Section VI. We also compare the results obtained in the new protocol with the results presented in [10]. Finally, Section VII gives some conclusions and future work.

II. The UUP Protocol

A. Cryptographic building blocks

In this section, some cryptographic blocks used in the UUP protocol are explained. Note that, some of them are also used in our proposal.

1) n-out-of-n threshold ElGamal encryption.: 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 it can only be decrypted if all n users collaborate in the process. Key generation, encryption and decryption processes 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 $\alpha = \alpha_1 + \cdots + \alpha_n$.
- *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:

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

- *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) 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. The resulting ciphertext corresponds to the same cleartext m.

B. UUP protocol overview

The UUP protocol includes a central node that receives requests from users. When it has n requests, it creates a new group. The users of this group build a group key (see Section II-A1). Then, every user encrypts her query with the group key and broadcasts it.

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

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

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

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

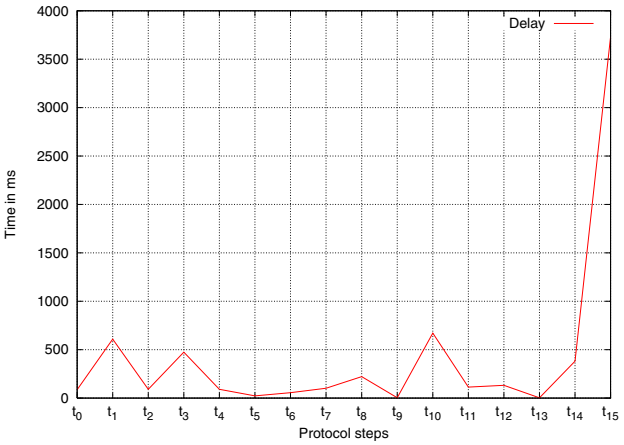

Fig. 1. Partial times of the *UUP protocol* in an open environment with $n = 3$ users and $l = 1024$ bits.

C. Computation and communication cost of the UUP protocol

In order to test this protocol, the authors of [10] performed several simulations. Different parameters were tested (*i.e.* number of users, key length, type of environment –local or open–). The authors indicated that a key of 1024 bits is computationally safe. They also argued that 3 users is the best value for the size of the group. Accordingly, this paper focuses on the tests they made in an open environment with $n = 3$ and $l = 1024$ bits.

However, two years have passed since [10] was tested. During this time, the Internet bandwidth and related resources have increased. Using the time results that appear on [10] could be detrimental to the UUP protocol in later comparisons. Therefore, it was necessary to simulate again the UUP protocol in the present conditions. This was done by executing the same source code that the authors of [10] used for their simulations. Figure 1 shows the obtained time results. Next, the meaning of each time interval is explained:

- t_0: time interval required to initialize the applet.
- t_1: time interval required by U_i to connect with the central node and get a response. This response includes the information needed to contact with the other members of the group and the parameters to create the group key.
- t_2: time interval required by U_i to create her group key share α_i.
- t_3: time interval required by U_i to establish a connection with the other group members.
- t_4: time interval required by U_i to broadcast her key share $y_i = g^{\alpha_i}$.
- t_5: time interval required by U_i to generate the group public key y using the shares received from all group members.
- t_6: time interval required by U_i to encrypt the query m_i using the group public key y.
- t_7: time interval required by U_i to send the resulting ciphertext c_i^0.
- t_8: time interval required by U_i to re-mask the received ciphertexts and permute them.
- t_9: time interval required by U_i to send the results which have been obtained in the re-masking/permuting step.
- t_{10}: time interval needed since user U_i sends her ciphertext c_i^0 until the last user broadcasts the ciphertexts $\{c_1, \cdots, c_n\}$. This period includes neither the time required to perform the re-masking/permuting step (t_8) nor the time required to send the ciphertexts to the next user (t_9).
- t_{11}: time interval required by U_i to calculate the shares which are used to decrypt the ciphertexts.
- t_{12}: time interval required by U_i to broadcast the shares.
- t_{13}: time interval required by U_i to decrypt the ciphertext c_i that she has received. Note that U_i needs all the shares sent by the other group users to perform this step.
- t_{14}: time interval required by the WSE to return the answer to the query m^i which has been sent by U_i.
- t_{15}: time interval required by U_i to distribute the received answer to the other users.

The increase of the Internet bandwidth and related resources benefits most of the time intervals. However, the time interval t_{15} has augmented significantly. This happens because, when the authors of [10] tested their protocol, a page of Google results took up about 50 Kilobytes. Nowadays, the size of a results page has increased up to 150 kilobytes due to the inclusion of multimedia content.

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

III. Our proposal in detail

A. Weak points of the UUP protocol

Figure 1 shows that $\{t_1, t_3, t_{10}, t_{15}\}$ are the time intervals which present the highest delay. Note that the most significant overhead is introduced by t_{15} (*i.e.* the broadcast of the results obtained from the WSE).

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

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

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

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

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

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

In the new protocol, before the broadcast phase, all users know the list of n queries which have been generated by the

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

In addition, with this modification users protect their privacy only if they follow the protocol. In [10], a selfish user who neither decrypts the ciphertexts, nor submits a query nor broadcasts the results can still get correct results if the rest of members of the group follow the protocol properly. As a result, users have no real incentive to behave honestly. On the other hand, in the new protocol, users preserve their privacy by hiding their own queries among queries of other users. In order to get the queries of the others, users must follow all the steps of the protocol.

B. Cryptographic building blocks

Some cryptographic building blocks in our protocol are the same as the blocks used in [10]. More specifically, the re-masking operation is performed using the *ElGamal re-masking technique* described in Section II-A2. Regarding the encryption, the new protocol employs the *n-out-of-n threshold ElGamal encryption* with the same *key generation* and the same *message encryption* (see Section II-A1). As stated before, the message decryption is performed using a new method named *message partial decryption* which is now explained:

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

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

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

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

IV. Protocol description

A. Group set up

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

B. Group key generation

1) Users $\{U_1, \ldots, U_n\}$ agree on a large prime p where $p = 2q+1$ and q is a prime too. Next, they pick an element $g \in \mathbb{Z}_q^*$ of order q.
2) In order to generate the group key, each user U_i performs the following steps:
   a) Generates a random number $a_i \in \mathbb{Z}_q^*$.
   b) Calculates her own share $y_i = g^{a_i} \bmod p$.
   c) Broadcasts her share y_i and receives the other shares y_j for $j = (1, \ldots, n)$, $j \neq i$.
   d) Uses the received shares to calculate the group key: $y = \prod_{1 \leq j \leq n} y_j = g^{a_1} \cdot g^{a_2} \cdot \ldots \cdot g^{a_n}$

C. Anonymous query retrieval

1) User U_i encrypts her query m_i:
   a) U_i generates a random number r_i.
   b) U_i encrypts her query m_i with the group key y: $c_i^0 = E_y(m_i, r_i) = (g^{r_i}, m_i \cdot y^{r_i}) = (c1_i, c2_i)$
2) For $i = (2, \ldots, n)$, each user U_i sends c_i^0 to the first member of the group (U_1).
3) For $i = (1, \ldots, n-1)$, each user U_i performs the following operations:
   a) Receives the list of ciphertexts $\left\{c_1^{i-1}, \ldots, c_n^{i-1}\right\}$.
   b) Using her share of the group key, partially decrypts the list of ciphertexts using the algorithm described in Section III-B. The resulting list of ciphertexts is denoted as $\left\{c_1^{i-1'}, \ldots, c_n^{i-1'}\right\}$.
   c) The list of ciphertexts $\left\{c_1^{i-1'}, \ldots, c_n^{i-1'}\right\}$ is re-masked using the re-masking algorithm described in Section II-A2 with a key $y' = \prod_{j=i+1}^{n} g^{\alpha_j}$. As a result, U_i obtains a re-encrypted version $\left\{e_1^{i-1}, \ldots, e_n^{i-1}\right\}$.
   d) Permutes the order of the ciphertexts at random, obtaining a reordered version $\left\{e_{\sigma(1)}^{i-1}, \ldots, e_{\sigma(n)}^{i-1}\right\}$
   e) Sends the list of ciphertexts $\left\{c_1^i, \ldots, c_n^i\right\} = \left\{e_{\sigma(1)}^{i-1}, \ldots, e_{\sigma(n)}^{i-1}\right\}$ to U_{i+1}.
4) The last user U_n performs the following operations:
   a) Receives the list of ciphertexts $\left\{c_1^{i-1}, \ldots, c_n^{i-1}\right\}$.
   b) Using her share of the group key, partially decrypts the list of ciphertexts using the algorithm described in Section III-B. At this point, U_n owns the cleartexts of the queries.
   c) Broadcasts the queries to the rest of users $\{U_1, \ldots, U_{n-1}\}$.

D. Query submission and retrieval

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

V. Privacy analysis

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

A. Dishonest user

Similarly to [10], in order to guarantee the correctness of the process, our protocol assumes an scenario where the users follow the protocol and there are no collusions. Nevertheless, the work presented in [13] modifies the scenario of [10] introducing malicious internal users. More specifically, [13] indicates three attacks that malicious internal users can do in order to learn the queries of other participants. We next detail how the proposed protocol behaves against the three attacks and how to modify it to work in the scenario of [13]:

1) **Stage-skipping attack**: In this attack, the last user U_n remasks and permutes the original list of ciphertexts instead of the list received from U_{n-1}. After the decryption, U_n knows which query belongs to each user. This attack was a threat in the UUP protocol, but it cannot be conducted in our protocol. This happens because our protocol employs the partial decryption operation. The original list of ciphertexts is encrypted under the group key while the list received from U_{n-1} is only encrypted under the key of U_n. User U_n cannot remove the encryptions under the shares of the other members of the group. Consequently, U_n cannot obtain the cleartexts from the original list of ciphertexts.
2) **Input-replacement attack**: In this attack, the first user U_1 learns the query of one of her partners. Initially, U_1 receives the original list of ciphertexts and removes all the ciphertexts except the one that belongs to her victim. Then, U_1 replaces the removed ciphertexts with individually remasked copies of her victim's ciphertext (or with encryptions of keywords chosen by U_1). When the last user broadcasts the queries, U_1 is able to identify which query belongs to her victim.
   In order to avoid this attack, we propose introducing zero-knowledge proofs in our protocol. Every user would have to prove that her outputs correspond to re-ordered and remasked versions of her inputs. This can be done using zero-knowledge proofs like the ones proposed in [18].
3) **Targeted public-key attack**: The key generation of Section II-A1 requires that all the users publish their shares at the same time. Otherwise, a participant can use the knowledge of the other shares to build her own. In this attack, a certain user U_j builds her key $y_j = g^{\alpha_j} / \prod_{i=1}^{n-1} y_i = g^{\alpha_j - \alpha_1 - \cdots - \alpha_{n-1}}$. Then, if y_j is used to build the group key, the result will be $y = g^{\alpha_j}$ and, hence, U_j will know the private group key.
   A solution for this attack is to broadcast previous commitments to the shares. Before sending her share, user U_i broadcasts a commitment $h_i = \mathcal{H}(y_i)$, where $\mathcal{H}$ is a one-way function. In the next step, the users exchange their shares and check that $h_j = \mathcal{H}(y_j)$ for $j = (1, \ldots, n)$.

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

B. Dishonest central node

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

C. Dishonest web search engine

The objective of the WSE is to gather the queries of the users in order to build their profiles. However, when the proposed protocol is executed, the WSE cannot know if a certain query has been generated by the user who has submitted it. This happens because when a user U executes the protocol, she submits her query hidden among other $n-1$ queries. Note that the other $n-1$ queries are not generated by a machine but by other real users. This means that the WSE cannot employ a method such as [6] or [7] to distinguish human queries from automatically-generated ones. Therefore, the WSE can correctly select the query that belongs to U with a probability of $\frac{1}{n}$. Note that, in order to build a useful profile, it is not enough for the WSE to select correctly one query in one execution of the protocol. Since the probability of selecting always the correct query is very low, the result is that the WSE obtains a distorted profile of the user.

Nevertheless, this probability can increase if U submits several queries about the same subject. In each execution of the protocol, U hides her query among a group of n queries. These queries are generated by randomly chosen users and, hence, they are likely to be related to different subjects. As a result, after several executions, the WSE might be able to find the common subject in the queries submitted by U and build her profile. This situation can be avoided by periodically submitting queries about irrelevant subjects to U. TrackMeNot [5] follows a similar approach using machine-generated fake queries. [6], [7] prove that machine-generated queries can be distinguished from human queries. To solve this problem, the proposed scheme can use the different $n-1$ human queries which are gathered at each protocol execution. These queries can be stored in a database for later use.

VI. Simulations

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

A. Implementation of the protocol

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

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

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

B. Time measures

Several simulations were performed in order to test the implementation of the protocol. The parameter selection was done using the same configurations proposed in [10]. This allows a later comparison between the UUP protocol and the new scheme.

In order to minimize the query delay, the creation of the group must be quick. According to [10], 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 [10] conclude that $n = 3$ is the most realistic group size. As stated in [10], the probability of forming a group of $n = 3$ users in a hundredth of a second is close to 1.

Consequently, the group size for our simulations was $n = 3$ users, the key length used of the cryptographic operations was $l = 1024$ bits. The protocol was executed in an open environment (computers located at different places and connected through the Internet). Each computer executes a single user application in each simulation. The environment is the same as for the simulations of Figure 1.

The protocol was executed 1000 times by each user. The average delay measures obtained for these executions are shown in Figure 2.

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

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

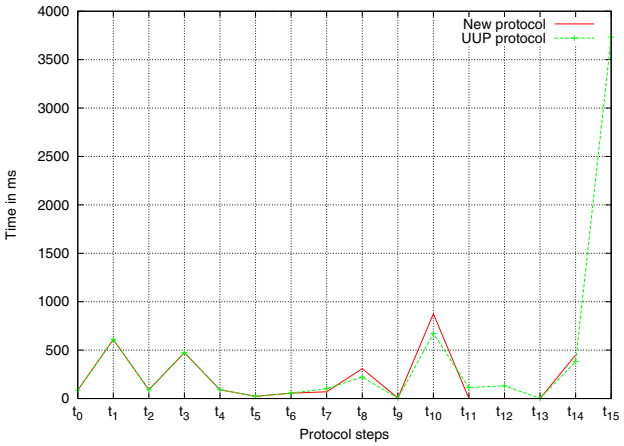

Fig. 2. Partial times for the simulations made in an open environment with $n = 3$ users and $l = 1024$ bits.

C. Comparison between the two protocols

The UUP protocol obtained a query delay of 5.2 seconds [10]. As explained in Section II-C, the size of the results page returned by Google has increased significantly in the last two years. Hence, nowadays, the expected query delay of this scheme would be 6.8 seconds approximately. The query delay achieved by the new protocol is 3.2 seconds. Therefore, the new proposal outperforms the results of [10]. It also outperforms the work presented in [12], which achieves the lowest delay (3.9 seconds) in the current literature. Note that this time was calculated using a simulated scenario. Thus, [12] should be tested in an open environment in order to be properly compared with the new proposal.

Figure 2 shows the different delays obtained by the UUP protocol and the new scheme. Next, the main changes in partial times are remarked:

1) Intervals $t_0 - t_6$ are almost equal in both protocols. Note that, t_4 includes the use of a hash function to prevent chosen public key attacks.
2) Since the number of exchanged messages is lower, the delay of t_7 is reduced. During this interval, in the UUP protocol the ciphertexts were broadcast (*i.e.* all the members had to send and receive n messages). In the new protocol all the users send *one* message, while the first user is the only who receives n messages.
3) The introduction of the new operation called *partial decryption* affects the time intervals $t_8 - t_{13}$. Because of this operation, the computational time required by the operations performed during t_8 has augmented. In addition, the time that a user has to wait to obtain the queries in cleartext (t_{10}) has also slightly increased. On the other hand, the time intervals $t_{11} - t_{13}$ disappear in the new scheme.
4) In the UUP protocol users only submit *one* query to the WSE. In the new protocol they submit n queries. Figure 2 shows the delay caused by submitting $n = 3$ queries. Note that, although this delay is higher than in the UUP protocol, it is not three times higher. This happens because the n queries can be submitted in parallel.
5) Figure 2 also shows that the UUP protocol has one step more than the new protocol. The longest time interval (t_{15} with more than 3.5 seconds) no longer appears in the new protocol. The removal of this delay is the most relevant improvement within the total time of the new scheme.

VII. Conclusions and future work

Web search engines have been proved to be a threat for the privacy of their users. Incidents like [2] and [3] show that users should not trust the companies behind the WSEs. Therefore, it is necessary to give to users a mechanism to prevent the WSEs from knowing their sensitive information.

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

In this paper, we propose a modification of the work presented in [10]. On one hand, our system optimizes some steps of the protocol to reduce the query delay. On the other hand, these changes incentivize every user to follow the protocol in order to protect their privacy. The new scheme has been tested in an open environment and the results show that it achieves the lowest query delay which has been reported in the literature.

However, it suffers from some practical and theoretical issues that need to be addressed. Firstly, the protocol must be adapted to work in a more hostile scenario. This means to efficiently prevent malicious internal attacks and collusions between entities. Secondly, the use of a central node to create the groups is a weak point and can represent a bottleneck. The future research should also focus on removing the central node from the protocol.

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] G. Conti, E. Sobiesk, "An honest man has nothing to fear: user perceptions on web-based information disclosure", *Proceedings of the 3rd symposium on usable privacy and security*, pp. 112–121, 2007.
[2] M. Barbaro, T. Zeller, "A Face is Exposed for AOL Searcher No. 4417749", *New York Times*, August 2006.
[3] K. Hafner, M. Richtel, "Google Resists U.S. Subpoena of Search Data", *New York Times*, January 2006.
[4] J. Domingo-Ferrer, A. Solanas, J. Castellà-Roca, "*h(k)*-private information retrieval from privacy-uncooperative queryable databases", *Journal of Online Information Review*, vol. 33, no. 4, pp. 1468–4527, 2009.
[5] TrackMeNot, 2011. `http://mrl.nyu.edu/dhowe/trackmenot`.
[6] R. Chow, P. Golle, "Faking contextual data for fun, profit, and privacy", *Proceedings of the 8th ACM workshop on Privacy in the electronic society – WPES'09*, pp. 105–108, 2009.
[7] S. T. Peddinti, N. Saxena, "On the privacy of web search based on query obfuscation: a case study of TrackMeNot", *Proceedings of the 10th international conference on Privacy enhancing technologies – PETS'10*, pp. 19–37, 2010.
[8] R. Dingledine, N. Mathewson, P. Syverson, "Tor: the second-generation onion router", *Proceedings of the 13th conference on USENIX Security Symposium*, pp. 21–21, 2004.
[9] F. Saint-Jean, A. Johnson, D. Boneh, J. Feigenbaum, "Private Web Search" *Proceedings of the 2007 ACM workshop on Privacy in electronic society – WPES'07*, pp. 84–90, 2007.
[10] J. Castellà-Roca, A. Viejo, J. Herrera-Joancomartí, "Preserving user's privacy in web search engines", *Computer Communications* vol. 32, no. 13-14, pp. 1541–1551, 2009.
[11] M. Reiter, A. Rubin, "Crowds: anonymity for Web transactions", *ACM Transactions on Information and System Security*, vol. 1, no. 1. pp. 66-92, 1998.
[12] A. Viejo, J. Castellà-Roca, "Using social networks to distort users' profiles generated by web search engines", *Computer Networks*, vol. 54, no. 9, pp. 1343–1357, 2010.
[13] Y, Lindell, E. Waisbard, "Private web search with malicious adversaries", *Proceedings of the 10th international conference on Privacy enhancing technologies – PETS'10*, pp. 220–235, 2010.
[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] G. Brassard, C. Crépeau and J.M. Robert, "All-or-nothing disclosure of secrets", *Advances in Cryptology – Crypto'86*, Lecture Notes in Computer Science, vol. 263, pp. 234-238, 1986.