Private Web Search with Constant Round Efficiency

Bolam Kang, Sung Cheol Goh and Myungsun Kim
Department of Information Security, The University of Suwon, Hwaseong, 445-743 South Korea

Keywords: Private web search (PWS), Secret sharing, Public-key encryption, Round efficiency.

Abstract: Web searches are increasingly becoming essential activites because they are often the most effective and convenient way of finding information. However, a web search can be a threat to the privacy of users because their queries may reveal sensitive information. Private web search (PWS) solutions allow users to find information on the Internet while preserving their privacy. According to their underlying technology, existing PWS solutions can be divided into three types: Proxy-based solutions, Obfuscation-based solutions, and Cryptography-based solutions. Among them, cryptography-based PWS (CB-PWS) systems are particularly interesting because they provide strong privacy guarantees.
In this paper, we present a constant-round CB-PWS protocol that preserves computational efficiency compared to known CB-PWS systems. To prove these arguments, we first analyze the efficiency of our protocol. According to our analysis, our protocol simply requires $3n$ modular exponentiations for n users. In particular, our protocol is a 5-round protocol that requires $O(n)$ communication complexity. In addition, evaluating the security of our protocol shows that our construction is comparable to similar solutions in terms of user privacy.

1 INTRODUCTION

A private web search (PWS) prevents web search service providers (e.g., Google and Yahoo) from building user profiles while still allowing users to enjoy the search functionality when performing web searches. User profiling is usually defined as the process of implicitly learning a user profile from search engine queries submitted by the user. Then, to perform user profiling, web search service providers use a user profile to classify a given user into predefined user segments (e.g., by demographics or tastes) or to capture the online behavior of the user, including the users private interests and preferences. This raises *privacy concerns* because sensitive information, such as a user's name and location, can be inferred from search engine queries. Aside from the query terms, other information such as the source IP address and timestamp may reveal sensitive information about the user.

Various approaches (e.g., (Elovici et al., 2006; Saint-Jean et al., 2007; Domingo-Ferrer et al., 2009; Lindell and Waisbard, 2010; Romero-Tris et al., 2011; Kim and Kim, 2012)) have been proposed to address this problem. In these systems, the main measure of efficiency is the *round complexity*, and it is important to construct constant-round PWS systems while guaranteeing privacy. In some cases, PWS schemes without strong privacy guarantees may suffice, and we know how to construct such protocols, e.g., using a proxy. However, in this work, we focus on cryptography-based PWS (CB-PWS) systems, where *strong privacy* is another important design goal.

To our knowledge, known CB-PWS constructions require $O(n)$ rounds, where n is the number of users (Castellà-Roca et al., 2009; Lindell and Waisbard, 2010; Romero-Tris et al., 2011), or significantly restrict the length of messages to be encrypted and hence do not lead to practical solutions to the problem (Kim and Kim, 2012). In addition, the latter requires web search engines to implement and run the protocols. Search engines, however, do not have any incentives to implement costly protocols that they cannot profit from. Unfortunately, there are no known constructions of practical constant-round CB-PWSs.

We briefly survey what is known in this regard about a private web search. We then summarize our contributions and provide a high-level overview of our construction.

1.1 Literature Review

Similar to (Balsa et al., 2012), we also believe that it is convenient to classify existing PWS solutions into

Kang B., Goh S. and Kim M..
Private Web Search with Constant Round Efficiency.
DOI: 10.5220/0005225602050212
In *Proceedings of the 1st International Conference on Information Systems Security and Privacy* (ICISSP-2015), pages 205-212
ISBN: 978-989-758-081-9

three categories. One category is based on a proxy server, called a proxy-base PWS solution; the second uses various randomizing techniques and is thus called an obfuscation-based PWS solution; and the third is a cryptography-based PWS solution. In the following, we provide somewhat detailed descriptions of each system. We begin with a proxy-based PWS scheme.

Proxy-based PWS: The first approach is through the use of an anonymous proxy (e.g., (Anonymizer, 2014; Scroogle, 2014; Saint-Jean et al., 2007)). Users can expect that anonymizers will prohibit the creation of user profiles through query unlinkability. There are several options in this category, from simple mechanisms achieving a low level of anonymity in web searches to more reliable but more complicate systems based on onion routing (Reed et al., 1998), such as the Tor network (Dingledine et al., 2004). However, the effectiveness of simple solutions is clearly limited. In addition, as highlighted by (Castellà-Roca et al., 2009), Tor cannot be installed and configured with relative ease. Further, it is well known that the HTTP requests over Tor can become very slow (Saint-Jean et al., 2007). For example, it takes 10 seconds on average to submit a query to Google even when using paths of length 2 (the default length is 3).

Obfuscation-based PWS: Another approach to providing privacy during web search is based on a query obfuscation technique (e.g., (Elovici et al., 2006; Domingo-Ferrer et al., 2009; Rebollo-Monedero and Forné, 2010; TracMeNot, 2014)). Roughly speaking, a class of solutions using query obfuscation involves blending the real queries into a stream of fake queries so that web search engines cannot create a correct profile. From a privacy point of view, these obfuscation-based solutions have a critical drawback: automated queries have features that are different from the actual queries entered by a user, such as randomness. The authors in (Peddinti and Saxena, 2010) demonstrated a concrete classifier that can distinguish real queries from fake queries generated by TrackMeNot (TracMeNot, 2014), with a mean misclassification rate of only approximately 0.02%.

Cryptography-based PWS: The last class of solutions involves using cryptographic algorithms, such as public-key encryption and shuffle. One of the main advantages of CB-PWSs over other approaches is that they provide strong privacy guarantees. In addition, they are not affected by the misclassification issue and are generally faster than anonymizer-based solutions. To our knowledge, known solutions can be found in (Castellà-Roca et al., 2009; Lindell and Waisbard, 2010; Romero-Tris et al., 2011; Kim and Kim, 2012). Without loss of generality, we may consider Romero-Tris et al.'s scheme as a malicious variant of Castellá-Roca et al.'s scheme. Thus, the difference between two schemes does not effect the round complexity.

The authors in the above references utilize the basic idea that upon joining a small-sized group, each user encrypts his search query and sends it to other members. Then, according to a predefined order, one user provides a shuffled list of encrypted queries to his neighbor. Finally, the last user broadcasts its shuffled version. After group decryption, each user obtains a set of queries, but he can not know who submitted which query. As a result, web search engines cannot build user profiles.

Further, we would like to note that the only approach that comes close to achieving our requirements in the restricted setting is the work by Kim et al. (Kim and Kim, 2012). The authors proposed a PWS scheme based on the notion of decomposable encryption. However, this approach significantly *restricts the length of plaintexts* (e.g., to 3 or 4 bits) to be encrypted and hence does not lead to practical solutions to the problem. Later, we provide a detailed evaluation and analysis of existing CB-PWS solutions (see Section 4).

1.2 Our Contributions

Our main contribution is that the first *practical* protocol has only $O(n)$ modular exponentiations and a *constant number of rounds* at the user side, where n is the number of users (i.e., the group size). According to our analysis of existing CB-PWS schemes (i.e., (Castellà-Roca et al., 2009; Lindell and Waisbard, 2010; Romero-Tris et al., 2011)), existing solutions require $O(n)$ computation complexity and $O(n)$ rounds under the same conditions.

Assume that a user has words or sentences for a query and is about to submit the query to a specific search engine. Using our protocol, the user first disperses his query term into n pieces using Shamir's secret sharing scheme. This can be very efficiently performed in a finite field (see Section 4.1). Then, encrypting each share into n ciphertexts under a public-key cryptosystem, which requires at most $O(n)$ modular exponentiations, each user broadcasts the encrypted shares to the corresponding users. Finally, all users send a list of re-masked and shuffled ciphertexts to a group manager. (In CB-PWS solutions, a small group of users is created and maintained by a specific entity called a group manager. We will explain the entity later.)

Our key technical contribution is distributing a query term instead of a private key among n users. Roughly speaking, known CB-PWS schemes demand that each user computes only a single ciphertext of the query term, but a collection of all n ciphertexts should be shuffled in a *relay manner* among all users. This step is essential to provide unlinkability between query terms and users, but it leads to $O(n)$ round complexity. Our scheme does not require users to participate in sequential shuffling.

1.3 Our Setting

We work in a setting consisting of the following three semi-honest entities:

- *Users*. The users are the individuals who submit query terms to the search engine and who wish to prevent the search engine from building their profiles. We use u to denote a user.
- *The group manager*. The role of the group manager, denoted by G, is to group users so that they execute our protocol that was introduced as above. We assume that the group manager has fairly powerful computing resources and storage capacity compared to the users.
- *The search engine*. The web search service provider, denoted by W, is the entity that provides a list of best-matching web pages, usually along with a short summary and/or, sometimes, parts of the document. A typical example is Google. Note that the search engine has no incentives to protect users' privacy.

We consider that an adversary is not allowed to break current computationally secure encryption schemes. We assume that there are at least two honest users. However, in contrast to (Castellà-Roca et al., 2009), we allow collusions between two entities of the protocol.

1.4 A High-level Overview of Our Solution

In what follows, we provide a high-level description of our construction and the techniques used therein. To obtain our construction, we build on the notion of secret sharing. The basic idea of Shamir's (t,n)-secret sharing is that a user can use a polynomial $f(X)$ of degree $t-1$ to split a secret q into n shares: $(\mathsf{v}_1,\ldots,\mathsf{v}_n)$. Then, any collection of shares $\geq t$ from the distributed n shares allows one to recover the secret using the Lagrange interpolation formula.

The starting point of the design of our solution is Shamir's secret sharing scheme: To submit a query term q, the user chooses a random polynomial $f(X)$ of degree $n-1$ whose constant term is q and evaluates $\mathsf{v}_i = f(u_i)$ at each user's label u_i. Then, each user computes encryptions $\bar{v}_i = \mathsf{E}_{pk}(\mathsf{v}_i)$ for all $1 \leq i \leq n$. Here, $\mathsf{E}_{pk}(\cdot)$ is a public-key encryption algorithm, and v_i is assumed to be in the message space of the encryption algorithm. Next, each user broadcasts all encrypted shares to other users.

After all users obtain a list of encrypted shares, they perform re-masking and permutation of the list and send it to a group manager without changing the original plaintexts of the ciphertext list. Using the homomorphic property of the underlying public-key encryption, we can efficiently and successfully re-encrypt ciphertexts. For the details of the homomorphism, see Section 2.2.

At first glance, our scheme may seem to be the same as existing CB-PWS schemes. However, we emphasize that in the above step of our scheme, each user has a list of ciphertexts that are different from all other users' lists, and thus, shuffling ciphertexts does not demand any interaction between neighbors. To the contrary, every user in existing solutions obtains the *same list of ciphertexts*. This requires every user to join in the sequential shuffles so that the protocol can achieve unlinkability. We consider this as a very legitimate reason to incur a high round complexity $O(n)$. However, our solution results in all users having *different lists of ciphertexts* so that the users do not need to perform shuffles sequentially.

To decrypt all received lists of ciphertexts, the group manager uses Lagrange interpolation to recover the users' query terms. The group manager then submits the recovered queries to the search engine and broadcasts the search results to the group users.

Outline of the Paper: This work is organized as follows. Section 2 introduces cryptographic building blocks: secret sharing and public-key encryption. Section 3 provides a detailed description of our construction. In Section 4, we provide the construction's performance and a security analysis.

2 BACKGROUND

In this section, we review the concepts and notation of cryptographic building blocks. We begin with a review of secret sharing techniques and then recall public-key cryptography and its security definitions.

Notation: For $n \in \mathbb{N}$, $[n]$ denotes the set $\{1,\ldots,n\}$. If A is a probabilistic polynomial-time (PPT) machine, we use $a \leftarrow A$ to denote making A produce an

output according to its internal randomness. In particular, if U is a set, then $r \xleftarrow{\$} U$ is used to denote sampling from the uniform distribution on U.

We denote by λ a security parameter. A function $g : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for every positive polynomial $\mu(\cdot)$, there is an integer N such that $g(n) < 1/\mu(n)$ for all $n > N$. We use standard asymptotic O notation to denote the growth of positive functions.

2.1 Secret Sharing

A secret sharing scheme is a method of distributing a secret, usually a key, among a group of users, requiring a cooperative effort to determine the key, so that the plaintext can subsequently be decrypted. The ultimate goal of the scheme is to divide the secret being hidden into n shares but whereby any subset of t shares can be used together to solve for the value of the secret. Additionally, any subset of $t-1$ shares will prevent the secret from being reconstructed. This is defined as a (t,n)-threshold scheme, meaning that the secret is dispersed into n overall pieces, with any t pieces being able to recreate the original secret.

In this work, we use Shamir's secret sharing scheme (Shamir, 1979). Shamir's scheme is based on polynomial interpolation and takes t points on the Cartesian plane; using those t points, a unique polynomial $f(X)$ is guaranteed to exist such that $f(X) = y$ for each of the points given. Regardless, this polynomial $f(X)$ is of degree $t-1$, and the coefficient for the 0^{th} degree is equal to a given secret q. Overall, the full equation for $f(X)$ is given as follows, with $\mathsf{q} = a_0$:

$$f(X) = a_0 + a_1 X + \cdots + a_{t-1} X^{t-1}.$$

2.2 Public-key Encryption

A public-key encryption scheme $\mathcal{E} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ consists of the following algorithms:

- KG is a randomized algorithm that takes a security parameter λ as input and outputs a secret key sk and a public key pk; pk defines a plaintext space $\mathcal{M}_{pk}$ and a ciphertext space $\mathcal{C}_{pk}$.
- E is a randomized algorithm that takes pk and a plaintext $m \in \mathcal{M}_{pk}$ as input and outputs a ciphertext $c \in \mathcal{C}_{pk}$. Note, this process is usually randomized using a randomization value $r \in \mathcal{R}_{pk}$

$$c = \mathsf{E}_{pk}(m; r)$$

- D takes sk and $c \in \mathcal{C}_{pk}$ as input and outputs the plaintext m.

We say that an encryption scheme is *correct* if, for any key-pair $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$ and any $m \in \mathcal{M}_{pk}$, $m \leftarrow \mathsf{D}_{sk}(\mathsf{E}_{pk}(m))$.

We say that a public-key cryptosystem $\mathcal{E} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ is *homomorphic* for the binary relations $(\oplus, \otimes)$ if for all $(pk, sk) \leftarrow \mathsf{KG}(1^\lambda)$:

- Given the message domain $\mathcal{M}_{pk}$, $(\mathcal{M}_{pk}, \oplus)$ forms a group.
- Given the ciphertext range $\mathcal{C}_{pk}$, $(\mathcal{C}_{pk}, \otimes)$ forms a group.
- For all $c_1, c_2 \in \mathcal{C}_{pk}$, $\mathsf{D}_{sk}(c_1 \otimes c_2) = \mathsf{D}_{sk}(c_1) \oplus \mathsf{D}_{sk}(c_2)$.

As a consequence, a cryptosystem's homomorphic property allows it to perform *reencryption*: given a ciphertext c, anyone can create a different ciphertext $\tilde{c}$ that encodes the same plaintext as c. Thus, given a homomorphic encryption scheme $\mathcal{E}$, we can define the reencryption algorithm as follows:

$$\mathsf{RE}_{pk}(c; r) = c \otimes \mathsf{E}_{pk}(m_0; r)$$

where m_0 is an identity message such that $\forall m \in \mathcal{M}_{pk}, m \oplus m_0 = m$. Further, $\mathsf{D}_{sk}(c) = m$, and then, $\mathsf{D}_{sk}(\mathsf{RE}_{pk}(c)) = m$, too.

Remark 1. *Herein, we will simply assume that we have some secret sharing scheme with the key K and a public-keycryptosystem with the key pk. The keys may or may not overlap, whereby elements from one key are also included in another key. For instance, we could imagine that the cryptosystem was an ElGamal scheme working on a group $\mathbb{G}$ of order q from primes $p, q | p-1$ and a generator g and that the secret sharing scheme used the same message space as the encryption scheme, i.e., $\mathcal{M}_K = \mathcal{M}_{pk}$.*

3 OUR CONSTRUCTION FOR PRIVATE WEB SEARCHING

We now describe our new technique for private web search in the semi-honest model. Our scheme enjoys a computational efficiency that is similar to existing CB-PWS schemes but does not require rounds in proportion to the number of users.

3.1 The Proposed Scheme

Our scheme is logically divided into three phases: *Setup*, *Mixing query*, and *Submitting query*. We provide an abstract description of our proposal in Figure 1.

Protocol: `Private_Web_Search`

– Inputs: For all $i \in [n]$, the input of u_i is a query term q_i.
– Auxiliary inputs: a security parameter 1^λ
– The protocol actions:
1. *Setup*. This step consists of two major tasks as follows:
   - Group setup: A group of n users is created, and the group information is given to the users by the group manager.
   - Parameter setup. The group manager generates the parameters for a public-key encryption scheme and a key for Shamir's secret sharing scheme and makes them publicly accessible to the users.
2. *Mixing query*. In this step, the users perform the following:
   (a) Splitting the query term q_i into n shares using Shamir's secret sharing scheme.
   (b) Encrypting the shares into a list of ciphertexts under the group manager's public key.
   (c) Broadcasting the list of ciphertexts.
   (d) Re-encrypting and mixing the list of ciphertexts.
   (e) Sending the list of ciphertexts to the group manager.
3. *Submitting query*. In this step, the group manager recovers all query terms by applying the Lagrange interpolation technique to the decrypted lists. Then, the group manager submits all query terms to the search engine.

Figure 1: A High-level Description of Our Protocol.

3.1.1 Setup

Let $\mathcal{E} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ be a semantically secure public-key cryptosystem with the homomorphic property, and let K be a public key specifying a list of parameters for Shamir's secret sharing scheme. It also specifies how to efficiently perform polynomial evaluation and interpolation (see Section 2.1). The Setup phase is composed of two primary activities:

1. When the group manager G receives n requests for private query, it responds to all n users by saying that the group size is n. Then, the group manager constructs a group $\{u_1, u_2, \ldots, u_n\}$ and publishes the group information. Specifically, the group information may include the group name, a list of participating users, and each user's label. We call this a group setup.
2. The group manager chooses a pair of keys (pk, sk) by invoking $(pk, sk) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$ and selects a proper key K with the message compatibility. It then publishes all system parameters (pk, K) for the protocol. As a result, the users have $\mathcal{M}_K = \mathcal{M}_{pk}$. We call this sub-step parameter setup.

3.1.2 Mixing Query

After obtaining the system parameters from G as a response to its query request, each user $u_{i\in[n]}$ performs the following:

1. Upon receiving the parameters, u_i chooses $n-1$ random coefficients $r_{i,j} \in \mathcal{M}_K$ and determines
$$R_i(X) = r_{i,n-1}X^{n-1} + \cdots + r_{i,1}X + \mathsf{q}_i$$
where $\mathsf{q}_i \in \mathcal{M}_K$ is u_i's query term.
2. Each user computes shares $\mathsf{v}_{i,j}$ of his query q_i for every $j \in [n]$ by
$$R_i(j) = \sum_{k=1}^{n-1} r_{i,k} j^k + \mathsf{q}_i.$$
We define $\mathsf{v}_{i,j} := R_i(j)$ for all $i, j \in [n]$.
3. For each $j \in [n]$, u_i computes $\bar{\mathsf{v}}_{i,j} = \mathsf{E}_{pk}(\mathsf{v}_{i,j})$ and broadcasts a list of ciphertexts $\langle i, j, \bar{\mathsf{v}}_{i,j}\rangle_{j\in[n]\setminus\{i\}}$ to all other users.
4. Assume that each user has the following list of encrypted shares
$$\begin{array}{ccccc} \perp & \cdots & \langle 1,i,\bar{\mathsf{v}}_{1,i}\rangle & \cdots & \langle 1,n,\bar{\mathsf{v}}_{1,n}\rangle \\ \langle 2,1,\bar{\mathsf{v}}_{2,1}\rangle & \cdots & \langle 2,i,\bar{\mathsf{v}}_{2,i}\rangle & \cdots & \langle 2,n,\bar{\mathsf{v}}_{2,n}\rangle \\ & & \vdots & & \\ \langle n,1,\bar{\mathsf{v}}_{n,1}\rangle & \cdots & \langle n,i,\bar{\mathsf{v}}_{n,i}\rangle & \cdots & \perp \end{array}$$
where $\perp$ indicates that a correct encrypted share is unknown to the corresponding cell.
In the above list, because the i-th column is complete, the user u_i sets $\bar{V}_i = (\bar{\mathsf{v}}_{1,i}, \ldots, \bar{\mathsf{v}}_{n,i})$. Then, the user computes a new version of the list $\bar{V}_i =$

$(\tilde{v}_{1,i},\ldots,\tilde{v}_{n,i})$, where $\tilde{v}_{\ell,i} = \mathsf{RE}_{pk}\left(\bar{v}_{\pi_i(j),i}\right)$ for all $\ell, j \in [n]$ and for a random permutation π_i over $[n]$.

5. Each user sends $\tilde{V}_i$ to the group manager.

3.1.3 Submitting Query

The group manager performs the following steps:

1. The group manager constructs the following $n \times n$ matrix $\mathfrak{M}$ by decrypting all of ciphertexts in the n vectors:

$$\mathfrak{M} = \begin{bmatrix} \tilde{V}_1 \\ \tilde{V}_2 \\ \vdots \\ \tilde{V}_n \end{bmatrix} = \begin{bmatrix} v_{\pi_1(1),1} & \cdots & v_{\pi_1(n),1} \\ v_{\pi_2(1),2} & \cdots & v_{\pi_2(n),2} \\ \vdots & \vdots & \vdots \\ v_{\pi_n(1),n} & \cdots & v_{\pi_n(n),n} \end{bmatrix}$$

2. Because G has no order information for each row, it sequentially recovers each query term by applying the Lagrange interpolation formula to each element in $\mathfrak{M}$.
3. The group manager submits a set of query terms to the search engine W. The group manager broadcasts the output from W.

Remark 2. *We have some options regarding the recovery of query terms. The simplest option that the group manager can choose is for the group manager to send all query terms reconstructed using the Lagrange interpolation formula in the Submitting Query step. This technique clearly ensures that all of the n original query terms are correctly determined. However, its major disadvantage is that it incurs high computation complexity.*

The alternative, albeit controversially, is to use a smart engine that can check if each reconstructed query term is in a dictionary maintained by the engine.

4 ANALYSIS

This section analyzes the performance of our construction in terms of the efficiency requirements. Next, we analyze the security of our protocol by examining all behaviors of the protocol.

4.1 Performance Analysis

Our protocol is compared to other CB-PWS solutions in terms of three efficiency measures: computation, communication and rounds. For this purpose, we first analyze the performance of our proposal. Then, the schemes proposed by (Castellà-Roca et al., 2009) and (Lindell and Waisbard, 2010) are compared to our proposal. We do not know how to provide a fair comparison between the scheme proposed by (Kim and Kim, 2012) and our scheme because our scheme allows query terms that are n-times longer than the maximum length of query terms allowed in (Kim and Kim, 2012).

Parameter Selection: Before conducting comparisons, we first need to determine the system parameters, the group size (n) and the key size.

We believe that the time users must wait to form the group determines the group size n. Quickly creating the group makes it possible to reduce the query delay. However, the larger the the group, the more privacy the protocol achieves. Therefore, one way that the members can obtain strong privacy is a scheme whereby a user joins a different small-sized group every time the user submits a query. According to (Castellà-Roca et al., 2009), $n = 3$ is the most realistic group size in practice. The authors in (Castellà-Roca et al., 2009) stated that, with an overwhelming probability, a group of $n = 3$ users can be created in a hundredth of a second.

Regarding the key length, we take a 1024-bit key length, as in other solutions. Thus, a 1024-bit key can encrypt up to 128 bytes at a time; therefore, a public-key cryptosystem that uses a 1024-bit key length can address queries of approximately 64 characters.

Computation Complexity: Now, we analyze the computation cost for running the protocol. In general, because it is widely accepted that modular exponentiations dominate the total computation cost of a system, we also focus on the number of modular exponentiations that every user must perform during execution of each protocol. For a fair comparison, we assume that our construction also employs an ElGamal encryption scheme.

For this purpose, we denote by $\mathsf{ME}(\ell)$ a modular exponentiation modulo of an ℓ-bit integer value. The scheme in (Lindell and Waisbard, 2010) extensively uses a double encryption by combining ElGamal encryption and Cramer and Shoup's cryptosystem (Cramer and Shoup, 1998). Thus, some modular exponentiations in (Lindell and Waisbard, 2010) should be carried out modulus a 2048-bit integer rather than a 1024-bit integer, as in (Castellà-Roca et al., 2009) and in our scheme. Our experimental implementation without any optimization shows that a 1024-bit modular exponentiation is approximately 10 times faster than a 2048-bit modular exponentiation. Specifically, a 1024-bit modular exponentiation

Table 1: Complexity Comparison at the User Side.

	Our scheme	Castellá-Roca et al.'s scheme	Lindell and Waisbard's scheme
Computation Complexity	$3n \cdot \mathsf{ME}(1024)$	$(3n+3) \cdot \mathsf{ME}(1024)$	$(n+3) \cdot \mathsf{ME}(1024) + 11n \cdot \mathsf{ME}(2048)$
Communication Complexity	$4n$	$3n-2$	$4n-2$
Round Complexity	5	$n+6$	$n+6$

takes 18 msec on average, whereas a 2048-bit modular exponentiation takes 191 msec on average.

We provide a comparison of schemes in the first row of Table 1 using the computation complexity. We remark that the evaluation of a polynomial in a finite field of degree less than n at n points can be performed using at most $O(\mathsf{M}(n)\log n)$ field operations, where $\mathsf{M}(n)$ is the number of bit operations for multiplying two n-bit integers. Similarly, Lagrange interpolation can be computed with the asymptotically same computation.

We observe that our scheme obtains the lowest computation cost from the user's point of view. Of course, our scheme and that in (Castellà-Roca et al., 2009) do not provide any mechanism to protect the honest users against malicious adversaries. However, even if we use zero-knowledge proofs to achieve active security, we conjecture that our scheme still has $O(n)$ computation complexity, as in (Lindell and Waisbard, 2010).

Remark 3. *In our scheme, the group manager has to perform somewhat heavy computations, while the group manager in other existing CB-PWS protocols only plays a role in maintaining a group of users. However, in practice, because n is small (e.g., $n=3$ or 4), we believe that these computations may not considerably affect the whole performance.*

Communication and Round Complexity: First, it is clear that our protocol only incurs a constant number of rounds. Next, we compare the communication complexity by counting the number of messages that every user should send in each step of the protocols. Table 1 summarizes the comparison results.

Our proposal consists of five rounds: (1) obtaining the system parameters, (2) applying Shamir's secret sharing and encryption to a query term and broadcasting a resulting list, (3) shuffling the resulting set and sending it to the group manager, (4) recovering a set of query terms and submitting to a search engine, and, finally, (5) broadcasting the search results to the users. All CB-PWS schemes have $O(n)$ communication complexity. In particular, our scheme only requires 5 rounds, while other CB-PWS proposals have $O(n)$ round complexity.

4.2 Security Analysis

Our construction achieves the following privacy requirements of the users when they submit query terms to a search engine:

Unlinkability Among Users: Let $J \subset [n]$ be a set of semi-honest users such that $|J| = \eta < n$. For notional convenience, suppose that $\hat{u} \in J$ is of the index $\alpha \in [n]$ but that it follows all steps of the protocol. At the end of the Mixing Query step, $\hat{u}$ receives a list of encrypted shares from other users: $(\bar{\mathsf{v}}_{1,\alpha}, \bar{\mathsf{v}}_{2,\alpha}, \ldots, \bar{\mathsf{v}}_{n,\alpha})$. If $\hat{u}$ would be able to decrypt these ciphertexts, it would be able to fill in some parts of the matrix $\mathfrak{M}$. Nevertheless, as long as $\mathfrak{M}$ is not completed, $\hat{u}$ cannot know the queries of honest users. For example, let u_1, u_2 be only the honest users. Denoting by $\boxtimes$ a decrypted share, $\hat{u}$ would be able to obtain at most the following matrix:

$$\begin{bmatrix} ? & \boxtimes & \boxtimes & \cdots & \boxtimes \\ \boxtimes & ? & \boxtimes & \cdots & \boxtimes \\ \boxtimes & \boxtimes & \boxtimes & \cdots & \boxtimes \\ & & \vdots & & \\ \boxtimes & \boxtimes & \boxtimes & \cdots & \boxtimes \end{bmatrix}$$

Hence, $\hat{u}$ is unable to obtain q_1 and q_2, and thus, our solution preserves the users' privacy.

Unlinkability between the Group Manager and the Users: Our protocol allows two entities (i.e., between $\hat{u}$ and G) to collude. In the middle of the Submitting Query phase, a compromised group manager and semi-honest users have the valid matrix, but they still do not know the secret permutations of honest users. Therefore, even though the group manager

would be compromised, the attacker could only randomly guess that a "link" is correct with a probability that is only negligibly greater than $\frac{1}{n-\eta}$.

Unlinkability between the Search Engine and the Users: Our protocol allows the search engine to participate in the execution of the protocol, except only at the end of the last phase. For reasons similar to those above, it cannot link a certain query to an honest user; therefore, it cannot build profiles for honest users.

5 CONCLUDING REMARKS AND FURTHER RESEARCH

In this work, we presented a constant-round CB-PWS protocol for protecting users' privacy. Our solution can be easily deployed in current systems because it does not require any changes on the service provider side. However, the following work remains for further research:

- We will try to provide a more rigorous security proof using standard techniques, such as simulation or game-playing proof. For this purpose, we first need to precisely define the notion of privacy in this setting.
- We should improve the performance of the group manager side, especially when the size of the group is large.

ACKNOWLEDGEMENTS

This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the Specialized Co-operation between industry and academic support program (NIPA-2014-H0808-14-1003) supervised by the NIPA(National IT Industry Promotion Agency). Myungsun Kim was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2014R1A1A2058377).

REFERENCES

Anonymizer (2014). Anonymizer. http://www.anonymizer.com.

Balsa, E., Troncoso, C., and Díaz, C. (2012). OB-PWS: Obfuscation-based private web search. In *IEEE Symposium on Security and Privacy*, pages 491–505.

Castellà-Roca, J., Viejo, A., and Herrera-Joancomartí, J. (2009). Preserving user's privacy in web search engines. *Computer Communications*, 32(13-14):1541–1551.

Cramer, R. and Shoup, V. (1998). A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk, H., editor, *Advances in Cryptology-Crypto*, LNCS 1462, pages 13–25.

Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The second-generation onion router. In Blaze, M., editor, *USENIX Security Symposium*, pages 303–320.

Domingo-Ferrer, J., Solanas, A., and Castellà-Roca, J. (2009). $h(k)$-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, 33(4):720–744.

Elovici, Y., Shapira, B., and Meshiach, A. (2006). Cluster-analysis attack against a private web solution (PRAW). *Online Information Review*, 30(6):624–643.

Kim, M. and Kim, J. (2012). Privacy-preserving web search. In *ICUFN*, pages 480–481.

Lindell, Y. and Waisbard, E. (2010). Private web search with malicious adversaries. In Atallah, M. and Hopper, N., editors, *Privacy Enhancing Technologies*, LNCS 6205, pages 220–235.

Peddinti, S. T. and Saxena, N. (2010). On the privacy of web search based on query obfuscation: A case study of TrackMeNot. In Atallah, M. and Hopper, N., editors, *Privacy Enhancing Technologies*, LNCS 6205, pages 19–37.

Rebollo-Monedero, D. and Forné, J. (2010). Optimized query forgery for private information retrieval. *IEEE Transactions on Information Theory*, 56(9):4631–4642.

Reed, M., Syverson, P., and Goldschlag, D. (1998). Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications*, 16(4):482–494.

Romero-Tris, C., Castellà-Roca, J., and Viejo, A. (2011). Multi-party private web search with untrusted partners. In Rajarajan, M., Piper, F., Wang, H., and Kesidis, G., editors, *SecureComm*, pages 261–280.

Saint-Jean, F., Johnson, A., Boneh, D., and Feigenbaum, J. (2007). Private web search. In Ning, P. and Yu, T., editors, *WPES*, pages 84–90.

Scroogle (2014). Scroogle, http://scroogle.org.

Shamir, A. (1979). How to share a secret. *Communications of the ACM*, 22(11):612–613.

TracMeNot (2014). TracMeNot, http://mrl.nyu.edu/dhowe/trackmenot.