

# Protecting Source Privacy in Federated Search

Wei Jiang      Luo Si      Jing Li

Dept. of Computer Science  
Purdue University

{wjiang, lsi, li66}@cs.purdue.edu

## ABSTRACT

Many information sources contain information that can only be accessed through search-specific search engines. Federated search provides search solutions of this type of hidden information that cannot be searched by conventional search engines. In many scenarios of federated search, such as the search among health care providers or among intelligence agencies, an individual information source does not want to disclose the source of the search results to users or other sources. Therefore, this paper proposes a two-step federated search protocol that protects the *privacy* of information sources. As far as we know, this is the first attempt to address the research problem of protecting source privacy in federated text search.

## 1. Introduction

Information plays crucial roles in today's fast pace-driven environment. How to find relevant information has been investigated in the field of information retrieval. Information can be copied (or crawled) and accessed by conventional search engines like Google or Yahoo!. When information is hidden from conventional engines, techniques applied to conventional search engine cannot be utilized directly. Federated search [1] has emerged as an answer to this problem.

Imagine the following two scenarios: 1) For better understanding of the spread of a specific disease, researchers may want to know if it has been reported by some health monitoring agencies. Due to possible liability, the agencies may not be willing to disclose their identities to the researchers. 2) Surgical accidents often occur across hospitals and health care centers. Reports are kept for the occurrences of such events. Some researchers may be interested in analyzing common mistakes among hospitals in a specific region so that more effective means to prevent these mistakes could be developed. However, some hospitals may not be willing to share their reports because of possible damage on their reputations and obligations to the public. In order to benefit from these information, source privacy needs to be protected in the federated search environment, while traditional fed-

erated search techniques are not applicable here. We next investigate how to achieve this objective.

## 2. Proposed Protocol

The following notations are adopted extensively throughout the paper:

- $S = \{S_1, \dots, S_m\}$ : a collection of  $m$  sources;  $S_i$  represents a source or a collection (depends on the context).
- $u, Q_u$ :  $u$  is a user, and  $Q_u$  is a user query issued by  $u$ .
- $E_{pu}, D_{pr}$ : a public key encryption scheme (e.g., RSA [4]) with a public-private key pair  $(pu, pr)$ .
- $E_{k_i}$ : a private key encryption scheme (e.g., AES [2]) with some private key  $k_i$ .

The protocol consists of two sub-protocols: ranked list generation (RLG) and content retrieval (CR). The RLG protocol simulates the behavior of conventional search engines to generate a global ranked lists. Once a user receives the ranked list, the CR protocol can be used to retrieve the desired content one at a time. Before detailing the RLG protocol, we first describe two functions used as building blocks for RLG: *Gen\_List* and *Merge*. The *Gen\_List* function generates an encrypted ranked list  $L_i$  based on  $Q_u$ . Suppose  $pu$  is the public key of a user  $u$ ,  $k_i$  is the private key of source  $S_i$ ,  $t_{ij}$  is the snippet of document  $d_{ij}$  at  $S_i$  and  $l_{ij}$  is the corresponding hyper link. Let  $L_{ij}$  denote the  $j^{th}$  element in  $L_i$ , then  $L_{ij}$  is formatted as  $E_{pu}(k_{ij}) || E_{k_{ij}}(t_{ij}, E_{k_i}(l_{ij})) || \delta$ , where  $\delta$  indicates some similarity score. Given  $L_i$  and  $L_j$ , the *Merge* function returns a new ranked list containing the top  $K$  elements between  $L_i$  and  $L_j$  according to  $\delta$ .

Algorithm 1 highlights the key steps of the RLG protocol. At step 1, a user broadcasts his or her query to all the sources in a specific federated search environment. At steps 2-4, each source independently generates an encrypted ranked list. (For now, we assume all information sources use the same retrieval algorithm). At steps 5-8, the ranked lists are merged sequentially from  $S_1$  to  $S_m$ , and finally,  $S_m$  sends the global ranked list to the user. Since the snippets are encrypted by private keys, the snippets generated at a specific source are not disclosed to any other sources during the merging process. Once the user receives the global ranked list  $L_m$ , using his or her public key, the user can get all the keys used to encrypt the snippets and their corresponding encrypted links. Based on the contents of the snippets and scores, the user can decide which document to be retrieved via the CR protocol. Note that public key encryption is used here to transport private keys (this is one of the most common uses of a public key encryption system in practice).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '07, July 23–27, 2007, Amsterdam, The Netherlands.  
Copyright 2007 ACM 978-1-59593-597-7/07/0007 ...\$5.00.

---

**Algorithm 1** The RLG Protocol

---

```
1:  $u$  sends  $Q_u$  to  $S_1, \dots, S_m$ 
2: for all  $S_i \in S$  do
3:    $L_i \leftarrow \text{GenList}(S_i, Q_u, \mathcal{K})$ 
4: end for
5: for  $i \leftarrow 2$  to  $m$  do
6:    $S_{i-1}$  sends  $L_{i-1}$  to  $S_i$ 
7:    $L_i \leftarrow \text{Merge}(L_{i-1}, L_i, \mathcal{K})$ 
8: end for
9:  $S_m$  sends  $L_m$  to  $u$ 
```

---

---

**Algorithm 2** The CR Protocol

---

```
1:  $u$  sends  $E_k(l)$  to  $S_1$ 
2: if  $\text{Retri}(S_1, E_k(l)) \neq \text{NULL}$  then
3:    $S_1$  sets  $d \leftarrow E_{pu}(k'_1) || E_{k'_1}(c_1)$ 
4: else
5:    $S_1$  randomly generates  $d$ 
6: end if
7: for  $i \leftarrow 2$  to  $m$  do
8:    $S_{i-1}$  sends  $d, E_k(l)$  to  $S_i$ 
9:   if  $\text{Retri}(S_i, E_k(l)) \neq \text{NULL}$  then
10:     $S_i$  sets  $d \leftarrow E_{pu}(k'_i) || E_{k'_i}(c_i)$ 
11:   end if
12: end for
13:  $S_m$  sends  $d$  to  $u$ 
```

---

Algorithm 2 highlights the key steps of CR. Suppose  $E_k(l)$  is the encrypted link from which the user wants to retrieve the desired document. First, the user sends  $E_k(l)$  to  $S_1$ . At step 3, the  $\text{Retri}$  function returns  $\text{NULL}$  if the link cannot be decrypted. If  $S_1$  can properly decrypt  $E_k(l)$  (i.e.,  $\text{Retri}(\cdot) \neq \text{NULL}$ ), it can use  $l$  to obtain the desired document, denoted as  $c_1$ .  $S_1$  then encrypts  $c_1$  with a key  $k'_1$ , and  $k'_1$  is encrypted using the user's public key. At the end of this step,  $S_1$  passes the encrypted document to the next source. If  $S_1$  cannot decrypt  $E_k(l)$ ,  $S_1$  randomly generates a document (pseudo-document) and passes it to the next source. At steps 7-12, sources  $S_2, \dots, S_m$  perform similar operations as  $S_1$  does at steps 2-6, except that whenever  $\text{Retri}$  returns  $\text{NULL}$ , the source just passes the received document to the next source. At the end,  $S_m$  sends the encrypted document to the user. The user then decrypts to get the private key that allows the user to obtain the actual document. Note that only one source has the intended document, and CR returns one document only. Suppose both  $S_1$  and  $S_2$  do not have the document. If  $S_1$  does not pass a pseudo-document to  $S_2$ , then  $S_2$  can infer that  $S_1$  does not possess the document desired by the user. Therefore, the pseudo-document is utilized to prevent such inference attack.

When the number of sources is large, the CR protocol becomes inefficient because it needs to visit every available source. Next, we briefly discuss how to solve this issue. The basic idea is that when each source generates the local ranked list in the RLG protocol, it randomly selects a subset  $t$  sources in  $S$  to generate a retrieval path. Let  $pu_i$  denote the public key of  $S_i$  and  $pu$  denote the user's public key. Suppose  $t = 3$  and  $\{S_1, S_2, S_5\}$  is the selected subset by  $S_2$ , then the retrieval path can be constructed as:  $E_{pu}(S_1 || E_{pu_1}(S_2 || E_{pu_2}(S_5 || E_{pu_5}(\text{NULL}))))$ . The path is appended after the encrypted document link. When the document is chosen by the user, he or she can decrypt and

find the first source on the retrieval path and then send the encrypted link to the source. In the previous example, the user sends  $E_k(l)$  and  $E_{pu_1}(S_2 || E_{pu_2}(S_5 || E_{pu_5}(\text{NULL})))$  to  $S_1$ . After performing the computation specified by the CR protocol,  $S_1$  can decrypt and find the next source on the path. Then  $S_1$  sends local computation results,  $E_k(l)$  and  $E_{pu_2}(S_5 || E_{pu_5}(\text{NULL}))$  to  $S_2$ . Since  $S_5$  is the last source on the path,  $S_5$  directly sends the user the retrieved document.

At each source, the computation complexity of RLG is similar to any commonly used IR algorithm to retrieve the top  $\mathcal{K}$  documents in the local database. The communication complexity of RLG is bounded by  $O(c \cdot m)$ , where  $c$  is the constant communication between any two adjacent sources assuming the public key and snippet sizes are fixed across the system. The computation complexity of CR at each source is bounded by the time to encrypt a document, and the communication complexity of CR is bounded by  $O(d \cdot m)$ , where  $d$  is the size of the document.

### 3. Related Work

Anonymization protocols proposed in [3, 5] are not suitable to achieve our objective. First of all, in our problem domain, the user identity needs not to be protected. As a result, we do not need to anonymize the communication from the user to the sources. Secondly, the source privacy needs to be preserved when they collaboratively produce the global ranked list. Since anonymizing the communication among the sources alone does not prevent a source from observing the contents of other sources' local ranked lists, source privacy is violated. Nevertheless, if preventing the disclosure of sources' IP addresses is a part of our objective, the aforementioned anonymization protocols could be utilized together with our proposed protocols.

### 4. Conclusion / Future Work

In this paper, we have proposed a federated search protocol to protect source privacy. One issue we have yet to discuss is the need for score global normalization. One may argue that global normalization can lead to consistent similarity scores. However, global normalization may distort the distribution of local source's data. We will address this issue in the future. In addition, we will add a resource selection phase to increase source privacy by hiding similarity scores and to make the retrieval protocol more efficient.

### 5. References

- [1] W.B. Croft, editor. *Advances in Information Retrieval*, chapter Distributed Information Retrieval, J. Callan, pages 127–150. Kluwer Academic Publisher, 2000.
- [2] NIST. Advanced encryption standard (aes). Technical Report NIST Special Publication FIPS-197, National Institute of Standards and Technology, 2001.
- [3] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
- [4] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [5] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *Proceedings of the 18th Annual Symposium on Security and Privacy*, pages 44–54, Oakland, CA, May 1997.