Privacy Preserving Metric-based Ranked Search over Encrypted Cloud Data

Shiyu Ji
shiyu@cs.ucsb.edu

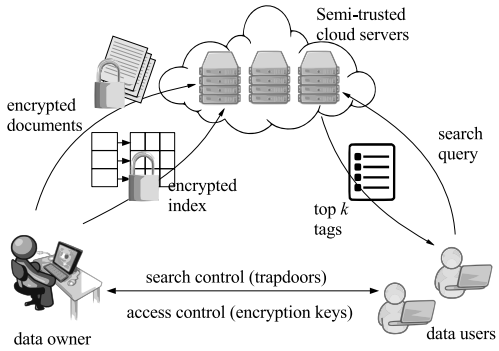

Fig. 1. The architecture of ranked similarity search over encrypted cloud data.

I. Introduction

With the advance of cloud computing and information retrieval, secure search over encrypted cloud data has attracted extensive research interest [1], [2], [3], [4], [5], [6], [7], [8], [9]. Most of the state-of-art research works focus on keyword relevance based search [1], [4], [7], [6], [8], or similarity based search [2], [3], [5], [9]. The measure of relevance or similarity is usually fixed, e.g., TF $\times$ IDF rule [1] or cosine similarity [2], [9], etc. It is still very open how to securely search over encrypted cloud data and rank the resulted documents based on various network metrics, e.g., PageRank [10], SimRank [11], Betweenness Centrality [12], Jaccard Index [13], etc. This paper attempts to give such a secure search framework supporting metric-based ranking.

We summarize our major contributions as follows:

1) We are the first to propose a privacy preserving scheme for all metric-based ranked search over encrypted cloud data.

II. Problem Formulation

A. The System and Threat Model

As in Fig. 1, we consider a cloud data hosting service which consists of three entities: the data owner, the data users and the semi-trusted cloud data server. The data owner has a collection of documents that need to be outsourced to the semi-trusted cloud server. The outsourced documents must be encrypted since the data owner does not want to reveal the documents to the semi-trusted server. The encryption scheme will be given later. On the other hand, the data users would like to search over the encrypted documents. To enable this, the data owner prepares a search index of the outsourced documents, and then uploads the encrypted index to the server. On a keyword-based search request, an authorized data user firstly gets the corresponding trapdoors and keys from the data owner, and uses the keys to generate the query given the keyword, and then sends the search query to the cloud server. Upon receiving the search query, the server should securely search on the index (i.e., without revealing any information about the documents), and return the tags of top k document (with highest scores based on some network metric) to the user. Finally the user uses the returned tags to securely obtain the documents from the cloud server, who should not know which documents are obtained. That is, the access pattern should be hidden.

We assume the cloud server is *semi-trusted*, or *honest-but-curious*, i.e., the server will honestly do as described above, but it also wants to know any information of the outsourced documents and the submitted search queries. We consider the *Ciphertext-Only Attack*, in which the server only has the submitted encrypted documents and queries, but it initially has no knowledge about their plaintext. We will also consider another stronger attack, in which we assume the server has some knowledge about the outsourced content, e.g., the frequency distribution. Thus some statistical attacks may be launched.

B. Our Goals

We want to design an efficient and secure ranked search scheme over encrypted cloud data that satisfies the requirements above, and we also want it to keep privacy to some extent.

Efficiency. The algorithms should be efficient, including: 1) encryption over the documents and index, 2) search over the encrypted index, 3) securely fetch the document given its tag, etc.

Dynamic updating support. The data owner should be able to update the documents and index efficiently at any time, e.g., insertion and deletion.

Confidentiality. The content of the documents, the entries (not yet queried) in the index and the queried keywords should not be revealed to the cloud server. We allow the cloud server to know the queried entries in the index since we would like to delegate the metric computation to the server for efficiency. Usually the computation will use the data in the queried entries.

Privacy preserving. We want to prevent the cloud server from learning anything about the outsourced documents, the keywords queried by users, or document access pattern. We will investigate the cases as follows:

- **Document privacy**. The server should not learn any information about the outsourced documents, e.g., which document was returned to the user.

- **Keyword privacy**. The server should not be able to know any information about the keywords in the search queries.
- **Oblivious document access**. The characteristic of document access pattern should not be learned by the cloud server. This can be achieved by using blind storage [14].

C. Notations and Definitions

- F — The collection of the documents to be outsourced: $F = \{f_1, \cdots, f_n\}$ where f_i denotes the i-th document in plaintext. Each document f_i contains several keywords $w_{i,1}$, $w_{i,2}$, etc.
- C — The collection of the encrypted documents in F: $C = \{c_1, \cdots, c_n\}$ where each c_i is the ciphertext of document f_i.
- W — The collection of all the keywords contained in the document F: $W = \{w_1, w_2, \cdots, w_m\}$ where w_i denotes the i-th keyword.
- I — The index built for the keywords W, where each index entry is a list containing information about the tags of documents C.
- r — The search request r, which contains the keywords which are interesting to the user.
- q — The query vector determined by the keywords in the search request r.
- T — The tags of top k documents: $T = \{\tau_1, \tau_2, \cdots, \tau_k\}$ with the highest scores based on the query q.
- $V[i]$ — The i-th component of the vector V. For example, if $V = (0, 1, 2)^T$, then $V[1] = 0$, $V[2] = 1$ and $V[3] = 2$.

Inverted Index is a popular technique in information retrieval [15]. An inverted index consists of multiple inverted lists, each of which corresponds to one keyword, i.e., each list of keyword w_i contains all the documents that contain the keyword w_i. Also the entries in the lists can include more information, such as the location of the document, the positions of the keyword within the document, etc. Inverted index outperforms most of other solutions when searching over very large volume of documents.

Graph model is one of the standard ways to define a network [16]. Usually the relations among the outsourced documents give a network, e.g., the citations, the references, etc. Many graph model based metrics have been proposed to evaluate the importance and centrality of each vertex in the network, e.g., PageRank, Betweenness Centrality, SimRank, Clustering Coefficient, etc. Most modern search engines rank the search results based on some network metric since users are often interested in the most keyword-related documents.

Blind storage [14] is an efficient oblivious transfer (OT) protocol implementation under the honest-but-curious adversary model. The locations of the stored documents are distributed independently from each other in the view of the cloud server. The cloud server will return the queried document correctly without knowing which one is returned. The communication and storage overhead of blind storage is moderate.

III. Basic Secure Ranked Search Scheme

In this section we propose our secure ranked search scheme, assuming the data owner can compute the metric scores of the documents before outsourcing them to the cloud server.

A. The Basic Scheme

The data owner randomly samples the symmetric encryption keys k, k' (given a security parameter λ) and chooses a pseudorandom function f. Let Enc denote the symmetric encryption and let $H(\cdot)$ be the collision resistant hash function.

- **Setup**: Given the document tags T and the keywords W, the data owner constructs the inverted index I. Denote by F_{w_i} the documents containing w_i and denote by T_{w_i} the tags of documents F_{w_i}. The data owner computes the metric scores of the documents in F_{w_i} and sort their tags T_{w_i} in descending order. The index I takes the keywords in W as keys, and takes the sorted T_{w_i} as the posting list of keyword w_i.
- **IndexBuild**: The data owner replaces each key w_i in the inverted index I by its hash value $H(w_i)$, and replaces its associated list T_{w_i} by the encryption $\mathsf{Enc}(T_{w_i}||r)$ under the key $f_k(w_i)$ (the value of f on input w_i under the key k), where $||$ denotes concatenation, and r is a random binary string that is used to hide the length information of T_{w_i}. Then the data owner outsources all the encrypted documents $\mathsf{Enc}_{k'}(F)$ and the encrypted inverted index I to the cloud server. The encrypted documents should be stored into a blind storage from the cloud server.
- **Retrieval**: Suppose the user would like to query the keyword w_j. With some help from the data owner, the user is able to compute $H(w_j)$ and $f_k(w_j)$. Then the user sends the tuple $(H(w_j), f_k(w_j))$ to the cloud server. Upon receiving the query, the server can locate the list by using $H(w_j)$ and decrypt the corresponding list $\mathsf{Enc}(T_{w_j}||r)$ by using $f_k(w_j)$. Then by restoring T_{w_j} in the list, the server will return the first K tags in T_{w_j} to the user.
- **OT**: Upon receiving the K top tags, the user communicates with the blind storage of the cloud server and safely obtains the corresponding K documents.

B. Security Analysis

Confidentiality of the documents and index entries (not yet queried) is guaranteed by symmetric encryption (e.g., 256-bit AES with CBC mode), which makes sure that it is computationally infeasible to distinguish its ciphertext from truly random strings.

Document privacy is protected by the blind storage, which assures that the cloud server has no idea which encrypted document was returned to the user.

Keyword privacy is protected by the collision resistant hash function $H(\cdot)$ and the pseudorandom function f, i.e., given $H(w_i)$ and $f_k(w_i)$ (the adversary has no idea on the key k), it is computationally infeasible to find the keyword w_i.

Oblivious document access is guaranteed by the blind storage scheme.

Note that the tags in the posting list of $H(w_i)$ are revealed to the cloud server once w_i is queried. Since in the blind storage scheme, tags are generated independently from the content of documents, exposing the tags will not negatively influence the document confidentiality or privacy. It is also possible to hide the tags from the cloud server: the user may send *only*

$H(w_i)$ to the server, and then the server returns the whole encrypted posting list of $H(w_i)$ to the user, and finally the user decrypts the list by using $f_k(w_i)$ and obliviously gets the top K files from the blind storage. However, by doing this the communication overhead can be quite high: the server has to return the whole list, which contains much information that is irrelevant to top K tags. Also since the user has to decrypt every time, the scheme will be less efficient.

C. Discussions

One drawback of the basic scheme is that it requires the data owner to pre-compute the metric scores of all the documents, which may be quite time-consuming. A natural idea is to delegate the computation to the cloud server, and the scores should be computed *only* when the lists are queried.

Also the basic scheme has to re-compute the metric scores and re-encrypt the corresponding lists when inserting or deleting documents. In our next improved scheme, this can be done more efficiently.

IV. Delegating Secure Ranked Search Scheme

In this section we propose our improved scheme that allows the data owner to safely delegate the metric computation to the cloud server. Our scheme can also allow the data owner to dynamically insert and delete documents to the cloud server efficiently.

A. The Delegating Scheme

Same as before, the data owner randomly samples the symmetric encryption keys k, k' and chooses a pseudorandom function f. The symmetric encryption Enc and the collision resistant hash function H are also chosen.

- **Setup**: Given the document tags T and the keywords W, the data owner constructs the inverted index I. Denote by F_{w_i} the documents containing w_i and denote by T_{w_i} the tags of documents F_{w_i}. Differently from before, the data owner does not compute the metric scores. Instead, for each tag $\tau_{i,j}$ in T_{w_i}, the data owner constructs a data structure with the fields as follows:
  1) Encryptions of *Tags* $\mathsf{Enc}(\tau_{i,j})$ under the key $f_k(w_i)$.
  2) Encryptions of *Edges* $\mathsf{Enc}(e(\tau_{i,j}, \tau_{i,k}))$ under the key $f_k(w_i)$ if among F_{w_i} there is an edge from the document $f_{i,j}$ to another $f_{i,k} \in F_{w_i}$ based on the graph model. The edge in the graph model can be interpreted as citation, reference, etc.

  Also to hide the length information of T_{w_i}, the data owner can generate some random data structures (denoted by strings r). Thus the inverted list of w_i is the collection of all the data structures defined above for each tag in T_{w_i}. We denote the list as L_{w_i}.
- **IndexBuild**: The data owner replaces each key w_i in the inverted index I by its hash value $H(w_i)$, and associates the above L_{w_i} as its associated list. Then the data owner outsources all the encrypted documents $\mathsf{Enc}_{k'}(F)$ and the encrypted inverted index I to the cloud server. The encrypted documents should be stored into a blind storage from the cloud server.
- **Retrieval**: Suppose the user would like to query the keyword w_j. With some help from the data owner, the user is able to compute $H(w_j)$ and $f_k(w_j)$. Then the user sends the tuple $(H(w_j), f_k(w_j))$ to the cloud server. Upon receiving the query, the server can locate the list by using $H(w_j)$ and decrypt the corresponding list L_{w_j} by using $f_k(w_j)$. Then the cloud server will learn the structure of the subgraph, in which each vertex denotes a document containing the keyword w_j. The cloud server can now compute (or approximate) the metric scores on the vertices, and select the tags with the K highest scores. Finally the server returns the top K tags to the user.
- **OT**: Upon receiving the K top tags, the user communicates with the blind storage of the cloud server and safely obtains the corresponding K documents.

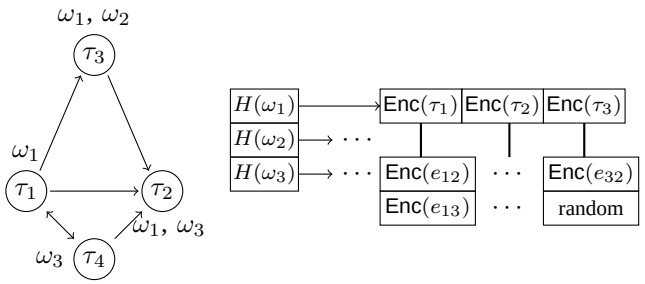

Fig. 2. A small example of the delegating scheme. There are four documents f_1, f_2, f_3, f_4 (with tags $\tau_1, \tau_2, \tau_3, \tau_4$) and three keywords $\omega_1, \omega_2, \omega_3$. In the inverted list of ω_1, there are three entries corresponding to the three documents which contain ω_1. Among these three documents, there are three edges: from τ_1 to both τ_2 and τ_3, and from τ_3 to $\tau 2$. Thus in the entry of $\mathsf{Enc}(\tau_1)$, we encrypt the edges e_{12} and e_{13}; in the entry of $\mathsf{Enc}(\tau_3)$, we encrypt the edge e_{32} (a random string is padded to hide the length information). The entry of $\mathsf{Enc}(\tau_2)$ encrypts no edge since τ_2 has no out-edge among the three documents containing ω_1. The entry of τ_2 may be padded with some random strings for privacy. Clearly with $H(\omega_1)$ and $f_k(\omega_1)$, the cloud server can recover the subgraph of τ_1, τ_2 and τ_3.

Fig. 2 gives a small example of our delegating scheme. Given any keyword, the cloud server can recover the subgraph, in which each node is associated with the given keyword. Then the server can compute the appropriate metric scores (e.g., PageRank, SimRank, Centrality, etc.) over the extracted subgraph, and return the top K tags based on the resulted scores. For example, on keyword ω_1, the server recovers the structure of the subgraph with nodes τ_1, τ_2 and τ_3. If the user is interested in the node containing ω_1 with the highest PageRank, then the cloud server will return τ_2, which is the most cited node in the PageRank sense.

B. Security Analysis

By a similar reasoning, document confidentiality and privacy, keyword privacy, oblivious document access pattern are all preserved by the scheme above. Since we want to delegate the metric score computation to the cloud server, we allow the server to know the subgraphs that were queried. The cloud server can maintain a cache to store the results. Then the server does not have to re-compute all the scores for each duplicated

query. Even though the server knows the subgraph structure upon query, the global network structure is still hidden. It is rare all keywords will be queried since in practice there is a significantly biased distribution on searched keywords. Moreover, since all the documents in the blind storage are encrypted by symmetric encryption, the server cannot infer the correspondence between the tags and the documents in plaintext.

C. Discussions

One major improvement of the scheme above is that now the data owner does not need to compute the metric scores, which saves a lot of time in the Setup phase. The tremendous computation work is delegated to the cloud server, which usually has high computing speed and abundant hardware resource. Also the computation of metrics follows a *lazy* pattern, i.e., they are computed only when needed. Since the distribution of queried keywords is usually highly biased, our improved scheme does not have to compute the metrics for all the documents or all the keywords in practice. Also it is clear that the improved scheme can be flexible to change network metric algorithms or policies, e.g., it is easy to change from PageRank to Betweenness Centrality.

Another important improvement is that we better support dynamic document insertion and deletion. Since each data structure is encrypted independently, the data owner can update them without refreshing the entire list.

REFERENCES

[1] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 23, no. 8, pp. 1467–1479, 2012.

[2] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems*, vol. 25, no. 1, pp. 222–233, 2014.

[3] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013, pp. 71–82.

[4] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 2112–2120.

[5] W. Sun, W. Lou, Y. T. Hou, and H. Li, "Privacy-preserving keyword search over encrypted data in cloud computing," in *Secure Cloud Computing*. Springer, 2014, pp. 189–212.

[6] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2110–2118.

[7] H. Li, D. Liu, Y. Dai, T. H. Luan, and X. S. Shen, "Enabling efficient multi-keyword ranked search over encrypted mobile cloud data through blind storage," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 127–138, 2015.

[8] B. Wang, W. Song, W. Lou, and Y. T. Hou, "Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee," in *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2015, pp. 2092–2100.

[9] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[10] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web." *Stanford Technical Report*, 1999.

[11] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 538–543.

[12] U. Brandes, "A faster algorithm for betweenness centrality*," *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[13] L. Hamers, Y. Hemeryck, G. Herweyers, M. Janssen, H. Keters, R. Rousseau, and A. Vanhoutte, "Similarity measures in scientometric research: the jaccard index versus salton's cosine formula," *Information Processing & Management*, vol. 25, no. 3, pp. 315–318, 1989.

[14] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 639–654.

[15] D. E. Knuth, "The art of computer programming, volume 1: fundamental algorithms. redwood city," 1997.

[16] M. Newman, *Networks: an introduction*. Oxford university press, 2010.