Efficient Privacy-Preservation Multi-factor Ranking with Approximate Search over Encrypted Big Cloud Data

Jing He[1], Yiming Wu[2], Guangli Xiang[3], Zhendong Wu[4], and Shouling Ji[2(✉)]

[1] Department of Computer Science, Kennesaw State University, 30060 Marietta, GA, Georgia
jhe4@kennesaw.edu
[2] School of Computer Science and Technology, Zhejiang University, Hanzhou, China
yiming96510@163.com, sji@zju.edu.cn
[3] School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China
glxiang@whut.edu.cn
[4] School of Cyberspace, Hangzhou Dianzi University, Hanzhou, China
wzd@hdu.edu.cn

Abstract. Encrypting data before outsourcing data has become a challenge in using traditional search algorithms. Many techniques have been proposed to cater the needs. However, as cloud service has a pay-as-you-go basis, these techniques are inefficiency. In this paper we attack the challenging problem by proposing an approximate multi keyword search with multi factor ranking over encrypted cloud data. Moreover, we establish strict privacy requirements and prove that the proposed scheme is secure in terms of privacy. To the best of our knowledge, we are the first who propose approximate matching technique on semantic search. Furthermore, to improve search efficiency, we consider multi-factor ranking technique to rank a query for documents. Through comprehensive experimental analysis combined with real world data, our proposed technique shows more efficiency and can retrieve more accurate results and meanwhile improve privacy by introducing randomness in query data.

1 Introduction

In order to provide on-demand access to resources, many companies are migrating services to cloud. However, there are huge amount of data in many places because the users can use them in remote location [1]. Researches have proposed some techniques on encrypting and outsourcing data into the cloud [2]. Given the expensive bandwidth cost, it's impossible to download and decrypt all the data locally, while we can search on encrypted data firstly and then download exact information. However, meeting requirements of performance such as accuracy, privacy and efficiency through this method can be quite challenging.

Particularly, we summarize our contribution of the paper as follows.

S. Wen et al. (Eds.): CSS 2017, LNCS 10581, pp. 452–459, 2017.
https://doi.org/10.1007/978-3-319-69471-9_33

- Using Stemming Algorithm for approximate matching to reduce search time complexity;
- Considering multiple factors to calculate approximating scores for ranking results more accurately;
- Efficient index construction by eliminating unimportant words to achieve optimized storage;
- Improving privacy-preservation by including dynamic dummy fields in the index and the query;
- Comparing experimental results with the state-of-the-art technique [3] in terms of searching time, accuracy, and privacy-preservation.

2 Problem Definition

2.1 System Model

The system model is illustrated in Fig. 1. In this paper, we use vector space model to model the documents stored in cloud server in which each data is represented as data vector when we refer data and document vectors. We explain the concept of multi keyword search in high dimensional space that has a particular data in dimensional matrix can be termed as data vector. In the context of multi keyword search algorithm, we model the data as dimensional matrix to perform ranking and retrieving ranking score for a particular query. Remember that we don't store index in matrix form but use binary tree index [3] technique to create index.

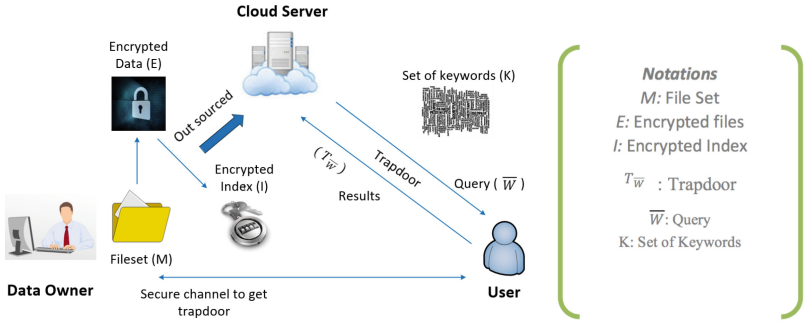

Fig. 1. System Architecture for search over encrypted data in cloud computing

2.2 Notations

In this section we summarize the notations used in the paper. We follow the same notations in entire paper:

- M: The plain text document collection denoted as set of n data documents M:$\{m_1, m_2 ...., m_n\}$. Where n is the number of files in the document collection.

- E: Encrypted documents stored in cloud server that can be denoted as E: $\{e_1, e_2, ...e_n\}$.
- W: The distinct keywords extracted from document collection M, denoted as $W = \{w_1, w_2, ..., w_n\}$.
- I: Searchable Encrypted index associated with E denoted as E:$\{i_1, i_2, i_3.i_n\}$, where each index i_i is built for each document m_i in document collection.
- $\overline{W}$: Represents keyword set in a search query that can also be considered as subset of W and denoted by $\overline{W}$:$\{\overline{w}_{j1}, \overline{w}_{j2}, \overline{w}_{j3}.....\overline{w}_{ji}\}$, where j represents the j^{th} keyword in the query, and i=1,2,3,n represents the i^{th} letter in j^{th} keyword.
- $T_{\overline{W}}$: The trapdoor generated for $\overline{W}$.
- $M_{id\overline{W}}$: The ranked *id list* of all documents for $\overline{W}$, where the subscript represents the id of the document retrieved for $\overline{W}$.
- K: Initial Keyword set created based on stem condition that can be represented as K: $\{k_1, k_2, k_m\}$ where m is the number of keywords in a the set.
- S_k: Secrete key generated by data owner used to encrypt, decrypt and to perform secure hash operations.
- T_1, T_2: They are the matrices to create secrete key.
- $\boldsymbol{V}$, $\boldsymbol{Q}$: Represents the data vector and query vector that can be multiplied to calculate to get the ranking score while searching.
- P_{score}: Score added to the term frequency rule by considering multiple factors.

3 Approximate Matching and Multi-factor Based Ranking Scheme

3.1 Approximate Matching

Trying to measure the similar between the created keywords with regard to the original query keyword, we propose *Approximation Score* to rank them. To be specific, the *Approximate Score* of k_i is denoted by *APP(*k_i*)*, can be formulate as follows:

$$APP(k_i) = Log_{10}((\frac{L_{ij}}{Q_j})) * ((\frac{U_{ij}}{Q_j})) \quad (1)$$

Algorithm 1. Create Stem Words

1: Procedure **createstem**($\overline{w}_j$, m, n) //$\overline{w}_j$represents single keyword in the given multiple keyword query;
2: **for** j = 0 to length of $\overline{w}_j$ **do**
3: **if** j $<$ m **then**
4: $p^{'} = +\overline{w}_{ji}$;
5: **else if** j $<$ (length($\overline{w}_j$) $- n$) **then**
6: $s^{'} = +\overline{w}_{ji}$;
7: **end if**
8: **end for**

The pseudo-codes of the whole process are summarized as follows. The stem words are created based on pre-defined conditions as shown in Algorithm 1. And then basic keyword sets are created on the basis of pre-created stem words (shown in Algorithm 2). Finally, the final keyword set is constructed by retrieving top l keywords in the basic keyword set (shown in Algorithm 3).

Algorithm 2. Basic Keyword Set Creation

```
Procedure keyword set traverse(I, p', S');
if w_ji==nil then
    return -1;
else
    return 1 + Math.max(height(I.root.left),height(I.root.right));
    //compute height of tree;
end if
if I.root == nil then
    return
end if
for i = 0 to height of tree do
    traverse(I.root.left, p', S');
end for
if I.root.left == p' then
    save in K_i;
else if I.root.left == S' then
    traverse(I.root.right, p', S');
end if
```

3.2 Ranking Similarity Measure

We consider two additional factors in calculating ranking score to further improve the precision. One the position of the keyword with respective to other documents and the other is the distance between the keywords in a sentence. The multi-factor based ranking score denoted by $Score(m_i,\overline{W})$ is formulated in Eq. 2.

$$Score(m_i, \overline{W}) = \frac{1}{|m_i|} \sum_{W_j \in W} (1 + ln(f_{m_i,w_j})) * ln(1 + \frac{|M|}{f_{w_j}}) + P_{score} \quad (2)$$

P_{score} is the padding score, which can be calculated by Eq. 3.

$$P_{score} = (\frac{f_{m_i,w_j}}{|m_i|})(1 - \frac{positon(w_j)}{|m_i|}) \quad (3)$$

4 System Framework and Efficient Search Mechanism

In this section we describe the basic framework of how our system works on encrypted data. The overview of the system framework is summarized in Fig. 2.

Algorithm 3. Final Keyword Set Creation

```
Procedure Set buildfinalset(K,l).//l represents the top keyword that needs to be searched;
if length of K <0 then
    Generate trapdoor on K;
else
    for i =0 to length of K do
        for j = 0 to j < i do
            if letter of i at j is matched with query of letter j then
                count = count+1;
            else if letter matched with position then
                temp = temp+1;
                Compute approximate score for each k_i;
            end if
        end for
    end for
    SORT(K); //keyword collection is sorted based on the approximate score;
end if
Return Set;
```

4.1 Random Dummy Field Insertion Mechanism

Trapdoor calculates the dummy fields with Eq. 4, and matches the keyword with the ranking score.

$$\begin{aligned} I_i * T_{\overline{W}} &= T_1^T * \overrightarrow{V_l}', T_2^T * \overrightarrow{V_l}'' * T^{-1}\overrightarrow{Q_l}', T_2^{-1} * \overrightarrow{Q_l}' \\ &= (\overrightarrow{V_l} * \overrightarrow{Q_l}) * (\overrightarrow{V_l}'' * \overrightarrow{Q_l}'') \Rightarrow V_i * Q_i \\ &= Score(m_i, \overline{W}) + \sum \mu^{(U)} + t \end{aligned} \tag{4}$$

4.2 Security Analysis

To provide privacy, dummy values are inserted by extending the dimension of data vector of trapdoor and the query. These random values can be inserted dynamically in the extended dimensions. Each time the trapdoor is generated dimension extensions can be different and random dummy fields would be different. Introducing dummy fields can produce different equations for each query. However, performance might could be compromised when introducing dummy fields in the extended dimension and differentiating from actual data vector. However, improving privacy is a tradeoff to loose performance in terms of computational speed and accuracy.

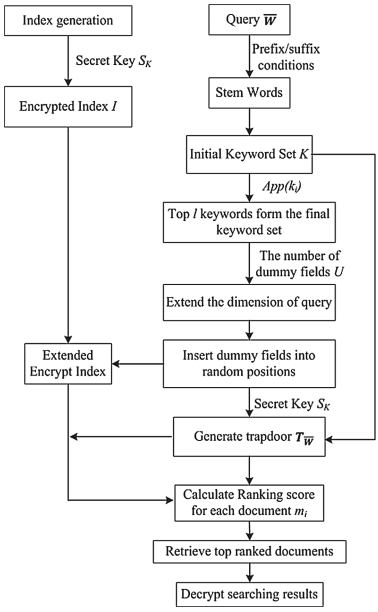

Fig. 2. Approximate searching process.

5 Performance Analysis

5.1 Simulation Settings

We build our own simulator through J2EE to simulate the cloud scenario. We used *Enron dataset* [4] that contains email information taken from *60000* users and randomly select subsets to form our testing dataset. Experiments are done for *700* files in a dataset and having *800* keywords in each file. To make smooth output, we averaged the values for every *100* instances. Three different schemes are implemented: (1) Our proposed scheme, noted by Approximation Search. (2) Privacy-Preserving Scheme in Known Ciphertext Model proposed in [5], noted by MRSE1. (3) Privacy-Preserving Scheme in Known Background Model proposed in [5], noted by MRSE2. We compare them in terms of the index creation time, trapdoor generation time, the query execution time.

5.2 Simulation Results

Index Construction Time. Index is created by extracting words from the document and each letter in the word forms a node. The index construction time includes the time to scan documents and create nodes in the index tree. Figure 3(a) shows the results of index construction time for all there algorithms.

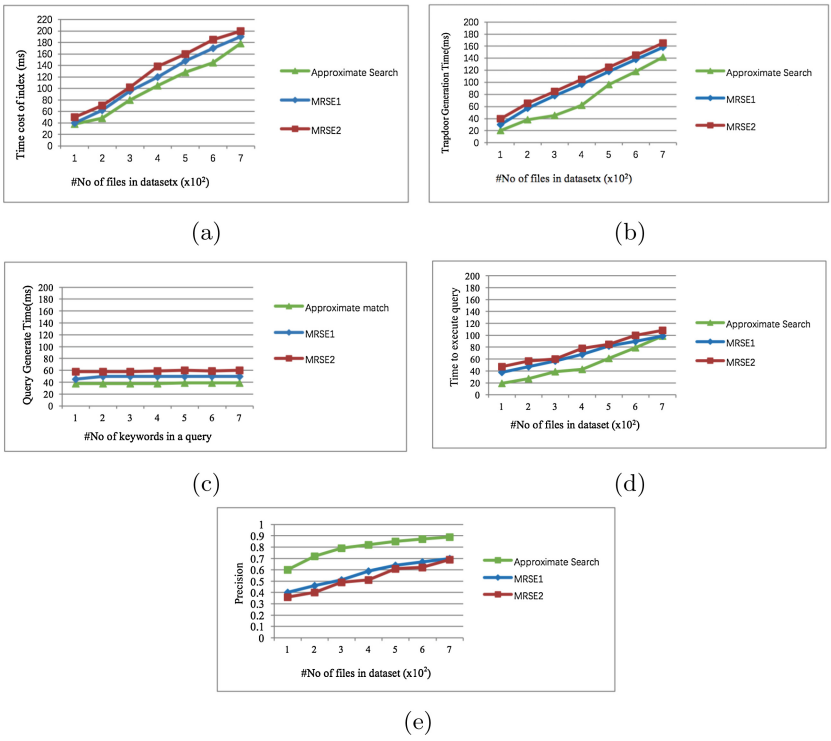

Fig. 3. Simulation results

Trapdoor Generation Time. Generating trapdoor requires the query and a secrete key. Figure 3(b) illustrates the time cost for trapdoor generation when a user gives a query to the server.

Query Generation and Execution Time. Query execution in server consists of creating and matching the hash values to differentiate the randomness from actual data, and ranking the order of the document. Figure 3(c) shows the results of query execution time with regard to the number of keywords in a query. Figure 3(d) shows the results of query execution time with regard to the number of files. Figure 3(e) summarizes the accuracy comparison results among three schemes.

6 Conclusion

In this paper we first introduce an approximate matching that can work on encrypted cloud data to improve searching efficiency. Subsequently, a multi-factor based ranking scoring technique is proposed to improve the accuracy of the searching results. Finally, to improve preserve privacy, a dynamic random

dummy value insertion scheme is proposed, so that it can withstand scale analysis attacks.

Acknowledgement. This work was partly supported by the Kennesaw State University College of Science and Mathematics the interdisciplinary Research Opportunities Program (IDROP), and the Office of the Vice President for Research (OVPR) Pilot/Seed Grant.

This was also partly supported by NSFC under No. 61772466, the Provincial Key Research and Development Program of Zhejiang, China under No. 2017C01055, the Fundamental Research Funds for the Central Universities, the Alibaba-Zhejiang University Joint Research Institute for Frontier Technologies (A.Z.F.T.) under Program No. XT622017000118, the CCF-Tencent Open Research Fund under No. AGR20160109, the National Key Research and Development Program of China (2016YFB0800201), and the Natural Science Fundation of Zhejiang Province (LY16F020016).

References

1. Chapman, C., Emmerich, W., Clayman, S.: Software architecture definition for on-demand cloud provisioning. Cluster Comput. **15**(2), 79–100 (2012)
2. Li, M., Yu, S., Cao, N., Lou, W.: Authorized private keyword search over encrypted data in cloud computing. In: 2011 31st International Conference on Distributed Computing Systems (ICDCS), pp. 383–392. IEEE (2011)
3. Ji, S., Li, W., He, J., Srivatsa, M., Beyah, R.: Poster: Optimization based data deanonymization2014. In: Poster Presented at the 35th IEEE Symposium on Security and Privacy, May, vol. 18, p. 21 (2014)
4. Wang, C., Wang, Q., Ren, K., Cao, N., Lou, W.: Toward secure and dependable storage services in cloud computing. IEEE Trans. Serv. Comput. **5**(2), 220–232 (2012)
5. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **25**(1), 222–233 (2014)