Future Generation Computer Systems 30 (2014) 179–190

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Efficient multi-keyword ranked query over encrypted data in cloud computing

Ruixuan Li [a,*], Zhiyong Xu [b], Wanshang Kang [a], Kin Choong Yow [c], Cheng-Zhong Xu [c,d]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China
[b] Department of Mathematics and Computer Science, Suffolk University, Boston, MA 02114, USA
[c] Shenzhen Institute of Advanced Technology, Chinese Academy of Science, Shenzhen, Guangdong 518055, China
[d] Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202, USA

HIGHLIGHTS

- Design a novel storage and encryption algorithm to manage the keyword dictionary.
- Greatly reduce both the dictionary reconstruction overhead and the file index re-encryption time as new keywords and files are added.
- Design a novel trapdoor generation algorithm.
- Take the keyword access frequencies into account when the system generates the ranked list of the returning results.

ARTICLE INFO

Article history:
Received 31 December 2012
Received in revised form
24 June 2013
Accepted 28 June 2013
Available online 17 July 2013

Keywords:
Cloud computing
Multi-keyword query
Ranked query
Top-*k* query
Data encryption
Privacy preserving

ABSTRACT

Cloud computing infrastructure is a promising new technology and greatly accelerates the development of large scale data storage, processing and distribution. However, security and privacy become major concerns when data owners outsource their private data onto public cloud servers that are not within their trusted management domains. To avoid information leakage, sensitive data have to be encrypted before uploading onto the cloud servers, which makes it a big challenge to support efficient keyword-based queries and rank the matching results on the encrypted data. Most current works only consider single keyword queries without appropriate ranking schemes. In the current multi-keyword ranked search approach, the keyword dictionary is static and cannot be extended easily when the number of keywords increases. Furthermore, it does not take the user behavior and keyword access frequency into account. For the query matching result which contains a large number of documents, the out-of-order ranking problem may occur. This makes it hard for the data consumer to find the subset that is most likely satisfying its requirements. In this paper, we propose a flexible multi-keyword query scheme, called MKQE to address the aforementioned drawbacks. MKQE greatly reduces the maintenance overhead during the keyword dictionary expansion. It takes keyword weights and user access history into consideration when generating the query result. Therefore, the documents that have higher access frequencies and that match closer to the users' access history get higher rankings in the matching result set. Our experiments show that MKQE presents superior performance over the current solutions.

1. Introduction

Cloud computing is getting more and more attention from both academic and industry communities as it becomes a major deployment platform of distributed applications, especially for large-scale data management systems. End users can outsource their personal data onto public clouds, and then access the data at anytime and anywhere. In the cloud environment, the resources allocated for each application can be scaled up and down according to the fluctuating demand. It adopts a pay-per-use resource sharing model, which allows a user to pay only for the number of service units it consumes. Cloud computing infrastructure provides a flexible and economic strategy for data management and resource sharing. It can reduce hardware, software costs and system maintenance overheads. It can also offer a convenient communication channel to share resources across data owners and data consumers. With the popularity of cloud services, such as Amazon Web Services,[1]

* Corresponding author.
E-mail address: rxli@hust.edu.cn (R. Li).

[1] Amazon web services, http://aws.amazon.com.

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.06.029

Microsoft Azure,[2] Apple iCloud,[3] Google AppEngine,[4] more and more companies are planning to move their data onto the cloud.

Despite of its advantages, the cloud computing infrastructure faces very challenging tasks, especially on data privacy, security and reliability issues. The fact is that private data are now placed on public clouds which are out of their trusted domains in cloud computing. Data owners do not have direct control over their sensitive data and are increasingly worrying about possible data loss and/or illegal use of their private data. Usually, cloud servers are considered as curious and untrusted entities. Data owners will hesitate to adopt cloud technologies if there are risks of data exposure to a third party or even the cloud service provider itself. Therefore, providing sufficient security and privacy protections on sensitive data is extremely important, especially for those applications dealing with health, financial and government data.

Some approaches have been proposed to evaluate cloud computing security and introduce a "trusted third party" to assure security characteristics within a cloud environment, such as [1,2]. To prevent information disclosure, the mainstream solution is to encrypt private data before uploading it onto the cloud server. On one hand, this approach ensures that the data are not visible to external users and cloud administrators. On the other hand, there are severe processing limitations on encrypted data. For example, standard plain text based searching algorithms are not applicable any more. To perform a keyword-based query, the entire data set has to be decrypted even if the matching result set is very small. It poses unbearable query latency and incurs unacceptable computational overhead.

To solve this issue, current solutions use the following strategy to provide keyword-based searching capabilities on encrypted data. First, a set of keywords are defined. An index vector is calculated for each file. It maintains the information of which keywords this file contains. After constructing the index vectors, an index file that combines all the index vectors is generated. The index file has to be encrypted as well. Second, both the encrypted data and index files are uploaded onto the data center servers in the cloud. Now, the data are ready to accept queries from the data consumers. The cloud servers can then support cipher text based queries as follows. A data consumer submits a keyword-based query, and the encrypted keywords are sent to the cloud server. The cloud server conducts a search on the encrypted index and returns a list of most relevant files. The user makes the decision that which files are needed and retrieves them from the server. After receiving encrypted files, the user decrypts the files with the associated key. This approach can guarantee the data security and preserve the data privacy. During the whole process, no plain text data or keywords are visible to the cloud servers.

Although substantial research works, such as [3–5], have been done to study keyword-based queries on encrypted data, many of them only address single keyword queries. Others use disjunctive or conjunctive searches for multi-keyword queries which have great limitations in flexibility and performance. Furthermore, few of them offer the ranking algorithm for matching results. MRSE [6] is the first and latest work to define such a multi-keyword ranked query problem, and proposes a viable solution to address it. In MRSE, all keywords are stored in a dictionary and a certain keyword can always be identified by its location in the dictionary. MRSE has two randomly generated invertible matrices for data and file index encryption operations. It uses the inner product of two vectors to build the trapdoor for secure keyword queries. It also applies an internal ranking algorithm to determine the top k files to be returned to the data consumer.

However, this approach suffers from three major drawbacks. First, it uses a static dictionary. If new keywords to be added, the dictionary has to be rebuilt completely which leads to substantial computational overhead. Second, an out-of-order problem occurs if using its trapdoor generation algorithm. Such a problem brings the result that the files with more matching keywords are likely excluded from the top k positions in the matching set. This means that the data consumer may not be able to find the most relevant files they want. Lastly, MRSE does not consider the effects of keyword weight and access frequencies. Therefore, the files that contain frequent keywords might not be included in the top k locations in the returning result at all.

In this paper, we design a new strategy called MKQE to address the aforementioned issues. In MKQE, we assume that the amount of data continues to increase from time to time. Accordingly, the keyword dictionary has to be expanded periodically. We propose a new dictionary construction paradigm, introduce a new trapdoor generation algorithm to reduce the query latencies, and take the keyword access frequencies into consideration to generate better matching result sets. In summary, we make the following contributions.

- We introduce partitioned matrices in the system design. The keyword dictionary can be expanded dynamically without touching the contents in the original dictionary. We design the novel storage and encryption algorithm to manage the keyword dictionary. MKQE greatly reduces both the dictionary reconstruction overhead and the file index re-encryption time as new keywords and files are added.
- We design a novel trapdoor generation algorithm. It can effectively reduce the impacts of dummy keywords on the ranking scores. With this new strategy, the out-of-order problem in the matching result set is solved.
- We take the keyword access frequencies into account when the system generates the ranked list of the returning results. Besides, we add the weights of the keywords in the index file. The files which contain more frequently accessed keywords will have higher weights in the query. The files with higher weights will have higher probabilities to appear in the first k locations of the matching result set. Hence, the data consumers have better chances to retrieve the desired files easily.

The rest of the paper is organized as follows. Section 2 defines the problem. Section 3 discusses current research works including MRSE and its drawbacks. Section 4 presents the system overview and the technical details of the proposed MKQE solution. Section 5 discusses the correctness and privacy-preserving analysis. Section 6 describes the experimental configurations and performance evaluations. Section 7 introduces the related works. Finally, Section 8 concludes the paper and gives the future work.

2. Problem definition

We aim to design a new approach to improve the performance for multi-keyword ranked queries on encrypted data in public cloud servers. In this section, we will introduce the notations and define the problem.

2.1. Notations

- F: the set of original files, assuming there are m files. F is denoted as $F = (F_1, F_2, F_3 \ldots F_m)$.
- C: the set of encrypted files, corresponding to the files in F. C is denoted as $C = (C_1, C_2, C_3 \ldots C_m)$.

[2] Microsoft azure, http://www.windowsazure.com.
[3] Apple icloud, https://www.icloud.com/.
[4] Google appengine, https://appengine.google.com/.