Volume 4, No. 11, November 2013
Journal of Global Research in Computer Science
RESEARCH PAPER
Available Online at www.jgrcs.info
ISSN-2229-371X

PRIVACY PRESERVING RANKED KEYWORD SEARCH OVER ENCRYPTED CLOUD DATA

Dinesh Nepolean, I.Karthik, Mu.Preethi, Rahul Goyal and M.K. Vanethi

Amrita Vishwa Vidyapeetham, Coimbatore, Tamilnadu, India
{ smnepolean, karganesh93, mupreethi,
goyal.1234rahul, vanethikathirvel } @gmail.com

Abstract: We present a scheme that discusses secure rank based keyword search over an encrypted cloud data. The data that has to be outsourced is encrypted using symmetric encryption algorithm for data confidentiality. The index file of the keyword set that has to be searched is outsourced to the local trusted server where the keyword set that is generated from the data files is also stored. This is done so that any un-trusted server cannot learn about the data with the help of the index formed. The index is created with the help of Aho-Corasick multiple strings matching algorithm which matches the pre-defined set of keywords with information in the data files to index them and store relevant data in B+ trees. Whenever the user searches for a keyword, the request is sent to the local trusted server and the indexed data is referred. The files are listed based on the certain relevance criteria. User requests for the required files to the un-trusted server. The parameters required for ranking is got from the data stored while indexing. Based on the ranking, the files are retrieved from the un-trusted server and displayed to the user. The proposed system can be extended to support Boolean search and Fuzzy keyword search techniques.

Keywords: Symmetric Encryption algorithm, Rank based search, multiple string matching, relevance scoring, privacy preserving, and cloud computing.

1. INTRODUCTION

Cloud Computing is the evolving technology that has changed the way of computing in IT Enterprise. It brings the software and data to the centralized data centers from where a large community of users can access information on pay per use basis. This poses security threats over the data stored. Data confidentiality may be compromised which has to be taken care of. So it becomes necessary to encrypt the data before outsourcing it to the cloud server. This makes data utilization a challenging task. Traditional searching mechanisms provide Boolean search to search over encrypted data, which is not applicable when the number of users and the number of data files stored in the cloud is large. They also impose two major issues, one being the post-processing that has to be done by the users to find the relevant document in need and the other is the network traffic that is undesirable in present scenario when all the files matching with keywords is retrieved. But this paper proposes ranked keyword search that overcomes these issues.

The paper is formulated as follows. The related work is summarized in Section 2. The proposed system and architecture diagram is covered in Section 3. The scheme is split into encryption module, string matching module, indexing module and ranking module which are also discussed under Section 3. Section 4 gives the gist about future ideas and proposals.

2.RELATED WORKS

It is an important research problem to enable the cloud service provider to efficiently search for the keyword in encrypted files and provide user with efficient search result maintaining data privacy at the same time. We have researched on the following papers.

2.1 Practical Technique for Search over Encrypted Cloud Data

This paper discusses on sequential scanning search technique [1] that searches over encrypted data stored in cloud without losing data confidentiality. The technique is provably secure and isolates the query result whereby the server doesn't know anything other the search result. It also supports functionalities such as controlled searching by server, hidden query support for user which searches for a word without revealing it to the server. With searchable symmetric encryption [7] and pseudorandom sequence generating mechanisms that are secure, encrypted data can be effectively scanned and searched without losing data privacy. The scheme that is proposed is flexible that it can be further extended to support search queries that are combined with Boolean operators, proximity queries, queries that contain regular expression, checking for keyword presence and so on. But, in case of large documents and scenarios that demand huge volumes of storage, the technique has high time complexity.

2.2 Public Key Encryption with Keyword Search

Dan Boneh proposed a solution for searching over the cloud data that is encrypted using the Public key Crypto System [2]. The idea is to securely attach or tag the related keywords along with the each file. This will avoid the need to completely decrypt the file and save the time of scanning entire file to check if the keyword exists. The file is encrypted using a public key encryption algorithm [2] and

© JGRCS 2013, All Rights Reserved

the keywords are encrypted by PEKS algorithm. To retrieve the document containing keyword W, send only the Trapdoor (W) to server. He proposed two methods for construction of this scheme, one using the bilinear maps and other using Jacobi symbols. The problem with this scheme is that every tag of all the files has to be processed for finding the match.

2.3 Boolean Symmetric Searchable Encryption

Most of the techniques discussed so far focused only on single keyword matching but in real-time scenarios users may enter more than one word. Tarik Moataz came up with a solution to tackle such challenges of searching multiple keywords over the encrypted cloud data. The construction of Boolean Symmetric Searchable Encryption (BSSE) [11] is mainly based on the orthogonalization of the keyword field according to the Gram-Schmidt process. The basic Boolean operations are: the disjunction, the conjunction and the negation.

2.4 Fuzzy Keyword Search

The traditional searching techniques retrieve files based on exact keyword match only but Fuzzy keyword search technique extends this feature by supporting common typos and format inconsistencies that occurs when the user types the keywords. The data privacy that is maintained during exact keyword search is ensured when this method is used. Wild card based technique [4] is used to create efficient fuzzy keyword sets that are used for matching relevant documents. The keyword sets are created using Edit Distance algorithm that quantifies word similarity. These keyword sets reduce storage and representation overhead by eliminating the need to generate all fuzzy keywords, rather generating on similarity basis. The search result that is provided is based on a fuzzy keyword data set that is generated whenever the exact match search fails.

3.PROPOSED SYSTEM

We have proposed an efficient scheme which enables the Cloud Service Provider
(CSP) to determine the files that are related to the keywords searched by the user, rank them and send the most relevant files without knowing any information about the cloud. Our schema consists of three entities: Data owner, Un-trusted cloud server and local trusted server. The data owner is the one whose data is stored in cloud server and he is also authorized to search over his files. Cloud server is an un-trusted server which provides storage service where data owners store their documents in encrypted form. The trusted local server stores the index that is created for the files. The system architecture is shown in Fig 1. We assume that authorization of users and keys used for encryption are managed by the local trusted server.

Notations:

1. C (F1, F2, .., Fn) : Files to be uploaded in cloud server.
2. W (w1, w2, ..,wi) : Keywords extracted from C.

3.1 System Architecture

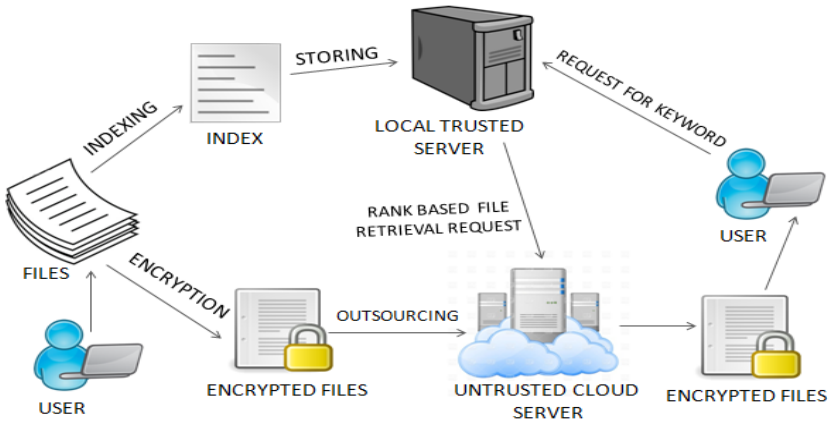

Fig. 1 System Architecture

3.2 Encryption Algorithm

As we are not going to perform any operation on the outsourced files to search of the keywords, we can use any of the existing light weight symmetric key Encryption algorithms and unload the data files to the cloud. We use DES to encrypt the file and then outsource it.

3.3 String Matching Algorithm

Aho-Corasick is found to be the efficient algorithm for multiple string matching that finds all occurrences of the pattern present in the files that are to be outsourced to the un-trusted cloud server.

The algorithm consists of two parts:

The first part is building of the tree (trie) from keywords we want to search for, and the second part is searching the test for the keywords using the previously built tree. The tree is a finite state machine, which is a deterministic model of

© JGRCS 2013, All Rights Reserved

behavior composed of finite number of states and transitions between those states. In the first phase of tree building, keywords are added to the tree where the root node is just a place holder and contains links to other letters. A trie is the keyword tree for a set of keywords K is a rooted tree T such that each edge of T is labeled by a character and any two edges out of a node have different labels.

CONSTRUCTION for set of keywords W = {W1, … Wk} and n = ∑= |Wi|.
Begin with the root node

Insert each keyword W, one after the other as follows:

Starting at the root, follow the path labeled by characters of $W_{i:}$

- If the path ends before Wi, continue it by adding new edges and nodes for the remaining characters of Wi
- Store identifier i of Wi at the terminal node of the path. This takes clearly O(|W1| + … + |Wk|) = O(n) time

LOOKUP of a string P:
Starting at root, follow the path labeled by characters of P as long as possible; If the path leads to a node with an identifier, P is a keyword. If the path terminates before P, the string is not in the set of keywords. This takes clearly O (|P|) time.

The files that are to be outsourced are given to the trie. Each word is looked up in the trie to check whether it is a keyword and the number of occurrences is stored. This value is then passed on to the next phase, which is Indexing.

3.4 Indexing

Index is created as a list of mappings [10] which correspond to each keyword. The list for a particular keyword contains details such as:

1. File ids of the files which has the particular keyword
2. Term frequency for each file which denotes the number of times the keyword has occurred in the file. This measures the importance of the keyword in that file.
3. Length of each file
4. Relevance score for each file
5. Number of files that has the particular keyword

Data structures such as B+ trees can be used to store this data. Term frequency, length of the file, number of files for the keyword are used to calculate the relevance score for each file by scoring mechanisms which is discussed later in the Ranking modules.

The previous papers discuss architectures [5][6] where both the index and the files are stored in encrypted form in the un-trusted server. Whenever user searches for a word, the request is sent to the un-trusted server, which searches over the index and sends the entire mapping that is created for the word to the user. The user has the overhead to decrypt and request to retrieve the most relevant files based on the relevance score information in the index. This takes up a huge amount of bandwidth and round trip time. To reduce the overheads, a new architecture that stores the index as plain text in the local trusted server is proposed. When user searches for a word, the word is sent to the local trusted server, which searches the index, finds out the most relevant files and requests un-trusted server for the files to be retrieved and sent to user thereby ensuring data confidentiality in un-trusted server.

Whenever a data file is stored, it is preprocessed to generate a index containing the aforesaid details using the keywords extracted (using multiple string matching algorithm discussed earlier) from the data file. The index creation scheme is as follows:

1. For each wi, that belongs to the keyword set W, generate F(wi) which denotes the file ids that contain wi
   2. For each wi € W
   For 1 <= j <= |F(wi)|
   2.1. Calculate score of the file Fij (with the help of scoring mechanisms discussed later) and store as Sij
2.2. Store it with file id id(Fij), length of the file |Fij| as (id(Fij) || |Fij| || Sij) in I(wi) which is the index list for the particular word wi
2.3. Update the total number of files that contain the keyword with the index list as (I(wi)||N)

The position of the word in the file is also considered for ranking the file. Hence the file that has the keyword in its title is considered to be more relevant than the files that has the keyword in their content. The relevance score is then stored in the index so that whenever the user requests for a word w, the top 'k' relevant files can be retrieved with this score.

3.5 Ranking

Once the documents are stored and indexed, the next important function is to rank them using details available such that the user retrieves the top ‘k’ most relevant documents. To do so, we need to calculate a numeric score for each file. In the IR community, the most widely used ranking functions are based on the TF X IDF rule, where TF stands for Term frequency which represents the number of times a keyword is present in a file and IDF stands for Inverse Document Frequency which is defined as the ratio of number of file containing the word to the total number of files present in the server.

The Ranking Function [5] used:

$$\text{Score}(W,Fi) = \sum 1/|Fi| \cdot (1 + \ln f_{i,t}) \cdot (1 + N/f_t)$$

W: Keyword whose score to be calculated
$f_{i,t}$: Frequency of term in file Fi
|Fi|: Length of the file
N : Total number of files in the collection.

4. CONCLUSIONS

In this paper, we solve the problem of post processing overhead and unnecessary network traffic created when Boolean search techniques are used, by introducing the

© JGRCS 2013, All Rights Reserved

ranked keyword search scheme. The scheme generates indexes that help the user to search for his documents in a secure environment. The files matching the keyword search are further ranked based on the relevant score calculated with term frequency, file length etc.

Further extensions to the project can be done by

1. Supporting multi user environment where there would be an extra entity in the scenario i.e., data user who is authorized to access other users files. The authorization mechanisms and key exchanges methods can be modified to support the same.
2. Tolerating minor typos and format inconsistencies that occur while typing the key words. This can be done by introducing fuzzy keyword mechanism discussed earlier.
3. Boolean Symmetric search technique can be included to support multiple keyword search without making any changes to the existing architecture.

5. REFERENCES

1. D. Song, D. Wagner, and A. Perrig.: "Practical Techniques for Searches on Encrypted Data." in Proc. of IEEE Symposium on Security and Privacy' (2000).
2. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano.: "Public Key Encryption with Keyword Search." in Proc. of EUROCRYP'04, volume 3027 of LNCS. Springer (2004).
4. Y.-C. Chang and M. Mitzenmacher.: "Privacy Preserving Keyword Searches on Remote Encrypted Data." in Proc. of ACNS'05 (2005).
5. J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou.: "Fuzzy Keyword Search over Encrypted Data in Cloud Computing." in Proc. of IEEE INFOCOM'10 Mini-Conference (2010).
6. 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 (2012).
7. C. Wang, N. Cao, J. Li, K. Ren, and W. Lou.: "Secure Ranked Keyword Search over Encrypted Cloud Data." in Proc. of ICDCS'10 (2010).
8. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky.: "Searchable Symmetric Encryption: Improved Definitions and Efficient Construction." in Proc. of ACM CCS'06 (2006).
9. Remya Rajan.: "Efficient and Privacy Preserving Multi User Keyword Search for Cloud Storage Services." International Journal of Advanced Technology And Engineering Research (IJATER), ISSN 2250 - 3536,Vol 2,Issue 4 (2012).
10. Zeeshan Ahmed Khan, R.K Pateriya.: "Multiple Pattern String Matching Methodologies" A Comparative Analysis (2012).
11. I. H. Witten, A. Moffat, and T. C. Bell.: "Managing Gigabytes: Compressing and Indexing Documents and Images." Morgan Kaufmann Publishing, San Francisco (1999).
12. Tarik Moataz and Abdullatif Shifka.: "Boolean Symmetric Searchable Encryption."

© JGRCS 2013, All Rights Reserved