

Secure Outsourced Text Data over Cloud Environment

¹ Manoj Gadekar, ² Sunilkumar Jaiswal

¹ Computer Science & Engineering, Dr. B.A.M. University, Jawaharlal Nehru Engineering college
Aurangabad, 431006, M.S., India

² Information Technology, Dr. B.A.M. University, Jawaharlal Nehru Engineering college
Aurangabad, 431006, India

Abstract - Cloud computing has become an integral part of IT industry. As it becomes prevalent, more and more sensitive information are being centralized in the cloud. A user stores his files in a cloud, and retrieves them whenever he wants to use them. The users search their documents through keyword in plaintext. There is a cloud service provider (CSP), whose purpose is to provide more and more security and privacy. To keep user data confidential against an unfrosted CSP, an easy way is to apply cryptographic approaches, by disclosing the data decryption key only to authorized users. Thus the keyword privacy is maintained. Although traditional searchable encryption techniques allow users to securely search over encrypted data through keywords, they support only Boolean search and are not yet sufficient to meet the effective data utilization need that is inherently demanded by large number of users and huge amount of data files in cloud. In this paper, we define and solve the problem of secure ranked keyword search over encrypted cloud data.

Keywords - *Ranked keyword Search, Symmetric Encryption algorithm, Rank based search, multiple string matching, relevance scoring, privacy preserving, and cloud computing.*

1. Introduction

Cloud computing provides a service for on demand users. Data has to be Encrypted before outsourced to the commercial public Cloud. Cloud Computing enables cloud customers to remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of computing resources. 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 are retrieved. But this paper proposes ranked keyword search that overcomes these issues.

2. Related Work

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 than 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 pseudo random 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 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.

2.3 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

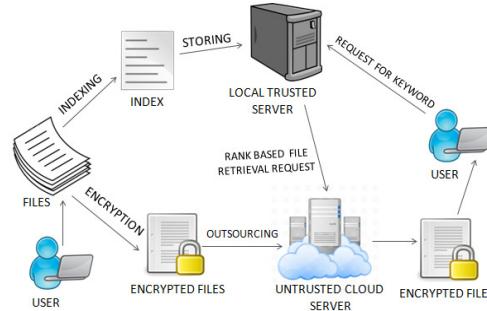


Fig. 1 System Architecture

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 AES to encrypt the file and then outsource it.

3.3 String Matching Algorithm

Aho-Corasick is found to be the efficient algorithm for multiple strings 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 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 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 $W = \{W_1, \dots, W_k\}$ and $n = \Sigma = |W_i|$.

Begin with the root node Insert each keyword W_i , one after the other as follows: Starting at the root, follow the path labeled by characters of W_i :

- If the path ends before W_i , continue it by adding new edges and nodes for the remaining characters of W_i
 - Store identifier i of W_i at the terminal node of the path. This takes clearly $O(|W_1| + \dots + |W_k|) = O(n)$ 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 w_i that belongs to the keyword set W , generate $F(w_i)$ which denotes the file ids that contain w_i
 2. For each $w_i \in W$ For $1 \leq j \leq |F(w_i)|$
 - 2.1 Calculate score of the file F_{ij} (with the help of scoring mechanisms discussed later) and store as S_i
 - 2.2 Store it with file id $id(F_{ij})$, length of the file $|F_{ij}|$ as $(id(F_{ij}) \parallel |F_{ij}| \parallel S_i)$ in $I(w_i)$ which is the index list for the particular word w_i
 - 2.3 Update the total number of files that contain the keyword with the index list as $(I(w_i))||N$

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 have 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 files containing the word to the total number of files present in the server.

The Ranking Function [5] used:

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

W: Keyword whose score to be calculated

$f_{i,t}$: Frequency of term in file F_i

|Fil|: Length of the file

N : Total number of files in the collection

4. Future Directions

There are possible improvements and undergoing efforts that will appear in the future work. This system is currently work on single cloud, In future it will extended up to sky computing & Provide better security in multi-user systems, the user side of proposed system will be implemented on mobile devices running Android and iOS operating systems since the potential application scenario envisions that users access the data anywhere and anytime. And the proposed method will be tested on a real dataset in order to compare the performance of our ranking method with the ranking methods used in plain datasets that do not involve any security or privacy-preserving techniques

5. Conclusion

We proposed the system to solve the problem of supporting efficient ranked keyword search for achieving effective utilization of remotely stored encrypted data in Cloud Computing. We first give a basic scheme and show that by following the same existing searchable encryption framework, it is very inefficient to achieve ranked search. We then appropriately weaken the security guarantee, resort to the newly developed crypto AES algorithm. The cloud setup consists of several virtual machines installed on it. Virtualization concept is the base for implementing cloud computing technology. Number of vm's can be deployed on the local host which completely depends on base system configuration. On each VM different types of server can be made to run. Data owner A selects any plaintext file from virtual machine (vm1) system which is to be uploaded on the cloud on another virtual machine (vm2). File chosen by the user A is encrypted by using AES encryption algorithm. The encrypted file is not readable to a normal user since that file contains cipher text in it. Authorized B user from local host can decrypt file, can access a confidential data.

Acknowledgments

We are thankful to the Department of Computer Science and Engineering for providing the Computer PG Lab at Jawaharlal Nehru Engineering College, Aurangabad (M.S.) India.

References

- [1] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling Secure and Efficient Ranked Keyword Search over Outsourced Cloud Data", Proc. IEEE, Parallel and Distributed Systems, Aug. 2012.
- [2] D. Song, D. Wagner, and A. Perrig.: "Practical Techniques for Searches on Encrypted Data." in Proc. of IEEE Symposium on Security and Privacy" (2000).
- [3] Y.-C. Chang and M. Mitzenmacher.: "Privacy Preserving Keyword Searches on Remote Encrypted Data." in Proc. of ACNS'05 (2005). [4] 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).

Manoj Gadekar completed his B.E. in Computer Science and Engineering from Dr. B.A.M. University Aurangabad. Currently he is pursuing his master's degree from same university. His research interest lies in cloud computing and data mining.

Prof. S. N. Jaiswal completed his BE (CSE) from Government College of Engineering, Aurangabad (1991), M. E. (CSE) (2009) from Walchand College of Engineering, Sangli. He has published 02 research papers in International Journals. His areas of interest are Data Mining and Cloud Computing