ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 7, September 2012

A SECURED SEARCHING IN CLOUD DATA USING CRYPTOGRAPHIC TECHNIQUE

C.BAGYALAKSHMI
Research scholar
Department of Computer Science
NGM College, Pollachi, Coimbatore, Tamil Nadu, India – 642001

DR.R.MANICKA CHEZIAN
Associate Professor
Department of Computer Science
NGM College, Pollachi, Coimbatore, Tamil Nadu, India – 642001

ABSTRACT

Cloud computing is the long dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. Trusted applications from untrusted components will be a major aspect of secure cloud computing. Today, many such applications require a proof of result correctness. It has been envisioned as the next generation architecture of IT Enterprise. As it becomes prevalent, more and more sensitive information are being centralized into the cloud. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. For the protection of data privacy, sensitive data usually have to be encrypted before outsourcing, which makes effective data utilization a very challenging task. With the advent of cloud computing, data owners are motivated to outsource their complex data management systems from local sites to commercial public cloud for great flexibility and economic savings. But for protecting data privacy, which obsoletes traditional data utilization based on plaintext keyword search. Multistep processing is commonly used for nearest neighbor (NN) and similarity search in applications involving high dimensional data and/or costly distance computations. Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords and selectively retrieve files of interest, these techniques support only exact keyword search. This paper explores various secured searching algorithms such as Advanced Encryption Standard (AES), Secured Socket Layer (SSL), and Knowledge nearest Neighbor (k-NN).

Index Terms - Distributed Hash Table, Multistep nearest neighbors, Index, Plain Text, and Cipher Text.

I Introduction

Cloud computing is the long dreamed vision of computing as a utility, where cloud customers can remotely store their data into the cloud so as to enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources. Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. Cloud computing provides three services are

SaaS (Software as a service), PaaS (Platform as a service), IaaS (Infrastructure as a service). The world of computation has changed from centralized to distributed systems and now we are getting back to the virtual centralization. A cloud computing system must make a copy of all its clients' information and store it on other devices. These copies are enabling to the central server to access backup machines to retrieve the data. Protecting privacy in cloud providers is a technical challenge. In cloud environment, this challenge is complicated by distributed nature of clouds and lack of subscriber knowledge over where the data is stored i.e. about data center and accessibility of the users.

Guided search protocol [10] to searching the files in the cloud network. In this Searching process first store the query history. If the search content is available in the cloud server then it will retrieve the content of the file from cloud. Else it forwards the query to its neighboring system, if the neighboring system has the search content, then it response to searching client. Else this process continues until last in the network, if no history is found then it forward to user.

Distributed Hash Table (DHT) [6] is a widely used building block for scalable cloud systems. However, as uniform hashing employed in DHTs destroys data locality, it is not a trivial task to support complex queries (e.g., range queries and k-nearest-neighbor queries) in DHT-based cloud systems. In order to support efficient processing of such complex queries, a popular solution is to build indexes on top of the DHT. Unfortunately, existing over-DHT indexing schemes suffer from either query inefficiency or high maintenance cost. In this paper, we propose light weight Hash Tree a query-efficient yet low-maintenance indexing scheme [3]. Light employs a novel naming mechanism and a tree summarization strategy for graceful distribution of its index structure. It's show through analysis that it can support various complex queries with near-optimal performance. Extensive experimental results also demonstrate that, compared with state of the art over-DHT indexing schemes, light saves 50-75 percent of index maintenance cost and substantially improves query performance in terms of both response time and bandwidth consumption. In addition, light is designed over generic DHTs and hence can be easily implemented and deployed in any DHT-based cloud system.

II PROBLEM DEFINITION

It can be classified into two categories: over-DHT indexing paradigm and overlay dependent indexing paradigm. While over-DHT indexing schemes treat data indexing as an independent problem free from the underlying cloud substrates, overlay-dependent indexing schemes are intended to closely couple indexes with the overlay substrates. Cloud users [12] are in unstructured cloud networks choose their neighbors and locally shared files using flooding techniques. In purely unstructured cloud networks such as blind search through flooding mechanisms is usually explored for resource discovery. To find a file, a cloud sends out a query to its neighbors on the overlay, until the query has travelled a certain radius. Despite its simplicity and robustness, flooding techniques, in general, do not scale. In large networks, the probability of a successful search may decrease dramatically without significantly enlarging the flooding radius.

It has lot of problems for cloud Search. Content distribution is a centralized one, where the content is distributed from the centralized server to all clients requesting the document. Clients send request to the centralized server for downloading the file. Server accepts the request and sends the file as response to the request.In most client-server setups [9], the server is a dedicated computer whose entire purpose is to distribute files.It will be solved all the above problems and increases the cloud

Search through Indexing. In blind search algorithm is used for searching files in the neighboring node which is inefficient and makes more time to search a file due to unawareness of the systems in the network. Future reference is not present in the routing table. [2] Routing update table is not present, which leads to delay in the search process and this makes the existing system inefficient. This system's provided the record-set as our proposed system but it should not be fully authenticated. Unable to use the public key cryptosystem, we have to search the nearest result accurately.

III SYSTEM MODELS

INDEXING MODEL

Search Indexer

It will recognize and creates an index of all the documents on the folders. The index is stored in a file called as the index file, where the search engine will find them.

Search Index File

Created by the Search Indexer program, this file stores the data from the files in a special index or database, designed for very quick access.

The indexing [7] procedure is based on the principle of choosing the set of adequate keys of a scalable size from the document collection, based only on document frequencies. The generated set of keys is based on selected terms and term sets that occur in at most *DFmax* documents, where *DFmax* is a parameter of our model. The crucial characteristic of this indexing method is that it leads to an increase in the total number of index entries, but, at the same time, limits the size of the associated posting lists to *DFmax*, which strictly bounds the traffic generated during retrieval. This approach is fully in line with the general of cloud networks that can easily store large amounts of data, but must be carefully controlled with respect to the volume of information transmitted between the client and server.

1. **Size filtering**

The size of an indexing key to a maximal size *Smax*, for example, in the case of file retrieval, the average query size is currently estimated to be between 2 and 3 terms, notice that limiting the size of the candidate keys does not have any substantial impact on the global indexing quality because, for a well chosen value of *Smax*, most of the user queries would have a size smaller than or equal to *Smax*. For a few queries of size bigger than *Smax*, a special retrieval mechanism is applied.

2. **Redundancy filtering**

This filtering method relies on the sub sumption property of the DKs to further reduce the number of candidate keys. If a key *k1* contains a discriminative key *k2* of a smaller size, then *k1* is also discriminative and the answer set of *k1* can be produced by local post processing of the posting list of *k2*. In other words, *k1* is practically redundant with respect to *k2* and therefore does not need to be stored in the global index. In other words, redundancy-based filtering implies considering only highly-discriminative keys and all their sub-keys for indexing. It greatly reduces the number of candidate keys, but, due to the sub summation property, fully preserves the indexing.

3. **Proximity filtering**

This filtering method uses the notion of textual context to reduce the number of generated keys. More precisely, we only keep keys containing terms that all appear in the same textual context, e.g., the same sentence, paragraph, or fixed-size document

window. The underlying argumentation is that words appearing close to each other in documents are good candidates to also count in queries. In our indexing model, we use the simplest textual context, a fixed-size window, and consider as keys only term sets exclusively consisting of terms occurring in a window of size *w*, where *w* is a parameter of our model.

DECENTRALIZED CLOUD EFFICIENT SEARCH MODEL

In this module we use guided search protocol to searching the files in the network. In this Searching process first its store the query history. If the search content is available in server then it will retrieve the content of the file from it. Else it forwards the query to its neighboring node. If the neighboring node have the search content, then it response to searching cloud data. Else this process continues in the network. If no history is found then it forward to source node. In case that the node chosen to forward the query already exists in the search history (hq), another node [8] with a relatively high probability of satisfying the query will be chosen instead. If all neighbor nodes have been visited before, then the node that enjoys the highest probability will be selected again for query processing.

The cloud server content distribution provides more resilience and higher availability through wide-scale replication of content at large numbers of nodes. A cloud content distribution community is a collection of intermittently-connected nodes with each node contributing storage, content and bandwidth to the rest of the community.The cloud server file sharing networks had a centralized server system. This system controls traffic amongst the users.

LOCAL TREE SUMMARIZATION MODEL

Need to map only leaf nodes to the underlying DHT. On the other hand, a bare leaf node lacks the knowledge of the overall tree structure, which, as we will see, is critical to complex query processing. Thus, we propose a distributed data structure, termed leaf bucket, to store data records and summarize the partition tree's structural information. Each leaf bucket corresponds to a leaf node in the tree. A bucket consists of two fields: leaf label, which maintains the label of the leaf node, and record store, which keeps all data records of the leaf node. For each leaf bucket, the label provides a local view of the partition tree, which is called a local tree. The local tree of leaf node #0100 consists of all its ancestors and their direct children. The label of any node in the local tree can be inferred directly from the label of each ancestor is a prefix of and the label of each branch node can be obtained by inverting the ending bit of a prefix. According to the completeness property of the partition tree, all branch nodes must exist in the tree. Some branch nodes may contain a sub tree, called the neighboring sub tree. The structures of these neighboring sub trees are unknown in the current local tree, but are maintained by some other leaves' local trees.

AES AND SSL ALGORITHM IMPLEMENTATION MODEL

Encryption is an important part. Robust encryption (AES over SSL) has been implemented and is available as an experimental. The secure communications modes include:

Integrity Protection

SSL/TLS protects against modification of messages.

Authentication

In most modes, SSL/TLS provides peer authentication. Servers are usually authenticated, and clients may be authenticated as requested by servers.

Confidentiality *(Privacy Protection)*

In most modes, SSL/TLS encrypts data being sent between client and server. This protects the confidentiality of data, so that passive message won't see sensitive data such as financial information or personal information of many kinds.

These kinds of protection are specified by a "cipher suite", which is a combination of cryptographic algorithms used by a given SSL connection. During the negotiation process, the two endpoints must agree on a cipher suite that is available in both environments. If there is no such suite in common, no SSL connection can be established, and no data can be exchanged. The SSL Socket class provides much of the same security functionality, but all of the inbound and outbound data is automatically transported using the underlying Socket, which by design uses a blocking model. While this is appropriate for many applications, this model does not provide the scalability required by large servers.

In this Searching process first its store the query history. If the search content is available in cloud server then it will retrieve the content of the file from local system. The file sharing networks had a centralized server system. This system controls traffic amongst the users. A distributed data structure termed leaf bucket, to store data records and summarize the partition tree's structural information.

Encryption [11] is an important part. Robust encryption (AES over SSL) has been implemented and is available as an experimental. These kinds of protection are specified by a "cipher suite", which is a combination of cryptographic algorithms used by a given SSL connection. During the negotiation process, the two endpoints must agree on a cipher suite that is available in both environments. If there is no such suite in common, no SSL connection can be established, and no data can be exchanged.

The SSL Socket class provides much of the same security functionality, but all of the inbound and outbound data is automatically transported using the underlying Socket, which by design uses a blocking model. While this is appropriate for many applications, this model does not provide the scalability required by large servers.

IV CONCLUSION & FUTURE SCOPE

These above works are based on the idea of space partitioning. Index structure into DHT in a straightforward manner, light leverages a clever naming function, which significantly lowers the maintenance cost and improves the DHT-lookup performance. Light employs local tree summarization to provide each bucket a local view. This local view is essentially helpful for distributed query processing, but unlike sequential leaf link, requires no extra maintenance cost, which retrieves distance information from distributed servers, eliminating those that cannot contribute results. In this paper, discuss about how retrieve data in distributed database. In future, it can be extended by implementing databases on distributed servers and thereby, reduce data replication. This is a private and secure place to share files and communicate with people. It is easy to use and free. File transfers use Bit Torrent-like technology. Files can be downloaded from multiple sources and uploaded to others at the same time.

REFERENCES

1. D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. of S&P*, 2000.
2. Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. of ACNS*, 2005.
3. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key

encryption with keyword search," in *Proc. of EUROCRYPT*, 2004.

4. 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*, San Diego, CA, USA, March 2010.
5. P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. of ACNS*, 2004, pp. 31–45.
6. J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. Of EUROCRYPT*, 2008
7. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Cryptography from anonymity," in *Proc. of FOCS*, 2006, pp. 239–248.
8. D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query Processing in Spatial Network Databases," Proc. Int'l Conf. Very Large Data Base Endowment (VLDB '03), 2003.
9. K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "Nearest Neighbor" Meaningful?," Proc. Int'l Conf. Database Theory (ICDT '99), 1999.
10. B. Waters, D. Balfanz, G. Durfee, and D. Smetters, "Building an encrypted and searchable audit log," in *Proc. of 11th Annual Network and Distributed System*, 2004.
11. M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proceedings of Crypto 2007, volume 4622 of LNCS*. Springer-Verlag, 2007.
12. D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. of EUROCRYP'04*, 2004.