A Multi-keyword Searchable Encryption Based on B+ Tree

Qin Chen, Shuyang Ye, Yalong Li, Min Zhang and Wangsheng Zhu

ABSTRACT

The increasing maturity of cloud computing promotes the development of searchable encryption technologies. In order to improve the retrieval efficiency, a Multi-keyword Searchable encryption technology based on the B+ tree is proposed in this paper. Establish indexes and retrieval trapdoor by using the space vector model and sort retrieved results according to the relevance score and the keyword matching. Experiments are carried out on real data sets, and the results show that the proposed scheme has high retrieval efficiency.

KEYWORDS

Cloud Server; B+ Index Tree; Searchable Encryption.

INTRODUCTION

Since the cloud service provider has strong ability of storage and computing, more and more users are willing to outsource their data to CSP, but the data owner loses the direct control over the data, facing the double threats of outside network attackers and trustless (honest and curious) managers in CSP. Researchers have proposed different solutions for different applications. Swaminathan et al. proposed a ranking search algorithm[1] in 2007. 2010, Cong et al. proposed a secure ranking
search algorithm[2]. Li and others based on keyword edit distance [3]. In recent years, researchers have proposed different solutions for different attack backgrounds and retrieval scenarios[4-9].

In order to improve the search efficiency, realizing rapid inquire without traversing every node in the tree. Major contributions of this paper are as follows:

1. Making keyword search priority of users according to their search and browse records.
2. Retrieval efficiency is improved by designing a multi-keyword searchable encryption scheme based on B+ tree retrieval.

PREPARATION

System Model

This multi-keyword searchable encryption based on preference model system consists of three aspects: data owner, authorized user, cloud server. As shown in Figure 1.

![Figure 1. Structure drawing of the encryption retrieval in the cloud environment.](image)

**B+ Retrieval Tree**

In it, document information is saved in leaf nodes, inner node saves the child information which is also the guidance node to help complete the retrieval process. B+ tree is a data structure which is balanced dynamically, and the information of each node is shown in formula (1):

\[ u = (id, fid, child[m], d[i]) \]  

(1)
Among which, $id$ is the number of node $u$, $fid$ means the identifier of the document. If $u$ is the inner node, then $fid=0$, $child[m]$ is the indicator pointing to the child node, $m$ is the order number of B+ tree; $d[i]$ is the keyword vector of the document which is calculated through formula (2):

$$d[i] = \max\{u.child[1] \rightarrow u.child[m] \rightarrow d[i]\}, i = 1, \ldots, m$$ (2)

**Construction of B+ Retrieval Tree**

After extracting the keyword and calculating the weight, each document should be transformed into space vector $\hat{q}$. All document information will be saved on the leaf node. The specific process is shown in Algorithm 1.

```
Algorithm 1: Insert(BNode u, fu)
if (fid != 0) then
    Insert fid into leaf node;
    update(d[i]);
    return;
else
    according to the new node id, find the child node $u'$;
    if (Num$_u'$>MaxNum) then
        Split $u'$, find the child node $u'$;
    end If
    Insert (BNode $u'$, fid);
end if
```

The weight of the keyword is saved in $d[i]$ of node $u$, if $d[i] \neq 0$, it means that in the sub-tree in which $u$ is taken as the root node, the leaf node to which at least one path points contains this keyword.

**SEARCHABLE ENCRYPTION BASED ON B+ TREE (BTSE)**

The four algorithms \{Setup, GenIndex(I), GenQuery($\hat{q}$), RelevanceScore(T,I)\}.
Setup: 1) A $m$-bit random vector $S$, 2) two ($m \times m$) random invertible matrices $M1$ and $M2$. $SK= \{ S, M1, M2 \}$.
GenIndex(I): first, construct unencrypted retrieval tree according to algorithm 1, and then encrypt the keyword weight in $\hat{d}$. Split $\hat{d}$ randomly into two vectors $\hat{d}_1$ and $\hat{d}_2$. If $S[i]=0$, $d[i]=d1[i]=d2[i]$, if $S[i]=1$, $d1[i]+d2[i]=d[i]$. Eventually, it is the two vectors $I= \{ \hat{d}_1, \hat{d}_2 \}$ that encrypt the nodes in the retrieval tree.
GenQuery( $\vec{q}$ ): The user inputs query to extract the keyword and generates query vector $\vec{q}$, similarly, encrypt the query vector according to key SK. If $S[i]=1$, $q[i]=q1[i]=q2[i]$, if $S[i]=0$, $q1[i]+q2[i]=q[i]$. At last, input trapdoor $T= \{ \vec{q}^1, \vec{q}^2 \}$.

RelevanceScore($T, I$): According to the trapdoor value and retrieval vector, the relevance score after the encryption is the same as the relevance score before the encryption. Therefore, top-k documents will be found which are the most relevant to the query. Then return them back to the user.

PERFORMANCE ANALYSIS

Index Tree Construction

Index generation consists of two steps: 1) Building B+ index tree based on document collection $F$; 2) Index tree is encrypted by two $(m \times m)$ invertible matrices. The overhead of encryption is directly dependent on the dimension of the data vector, mainly related to the size of the keyword dictionary and the document collection. In Figure 2 (a), the size of the keyword dictionary is $m=4000$, $k=100$. In Figure 2 (b), the size of the Chinese file set is $m=5000$, $k=100$. The overhead of the index tree is proportional to the number of documents and dictionary. As shown in Figure 2.

![Figure 2. Time cost of index tree construction.](image)
Search Efficiency

The search time include the time of relevance score calculation and finding the top-k documents. We obtain the time consumption under two different factors. In Figure 3 (a) the fixed keyword dictionary $m=4000$, $k=100$. With the increase of documents, the number of files that contain keywords to be retrieved is likely to increase, so the time is gradually increasing. Compared with MRES, BTSE have significant advantages in retrieval time, because the MRES scheme needs to traverse each document index with near linear computational complexity. In Figure 3 (b) with the fixed document collection, $n=5000$, $k=100$. As the number of keywords increases, the retrieval time increases gradually. This is because, with the increase of keywords, the dimension of the index vector is also increasing, and the time consumption is greater when the correlation calculation is done. As shown in Figure 3.

CONCLUSIONS

This paper uses B+ tree index structure. Compared with the binary tree, it reduces the spatial complexity of the retrieval and improved the retrieval efficiency. Besides, this paper creates a searchable encryption scheme based on the interest model according to historical actions when users browse documents. A large number of experiments verify that this scheme can provide more accurate results in meeting personalized searches of users. In the future, different encryption methods will be used to encrypt the retrieval so that the retrieval can be safer.
ACKNOWLEDGEMENTS

This work is supported by Zhejiang Provincial Technical Plan Project (No. 2017C01064, 2017C01055, 2018C01088).

REFERENCES