IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 10, 2018, accepted March 11, 2018, date of publication March 19, 2018, date of current version April 4, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2816919

Secure and Efficient Product Information Retrieval in Cloud Computing

YING-SI ZHAO[1] AND QING-AN ZENG[2], (Senior Member, IEEE)
[1]School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China
[2]Department of Computer Systems Technology, North Carolina A&T State University, Greensboro, NC 27411, USA

Corresponding author: Ying-Si Zhao (8769@bjtu.edu.cn)

This work was supported by the Fundamental Research Funds for the Central Universities under Grant 2015RC021.

ABSTRACT Cloud computing is a promising information technique (IT) that can organize a large amount of IT resources in an efficient and flexible manner. Increasingly numerous companies plan to move their local data management systems to the cloud and store and manage their product information on cloud servers. An accompanying challenge is how to protect the security of the commercially confidential data, while maintaining the ability to search the data. In this paper, a privacy-preserving data search scheme is proposed, that can support both the identifier-based and feature-based product searches. Specifically, two novel index trees are constructed and encrypted, that can be searched without knowing the plaintext data. Analysis and simulation results demonstrate the security and efficiency of our scheme.

INDEX TERMS Product information retrieval, cloud computing, information security.

I. INTRODUCTION

Driven by the revolution of information technology in recent years and with the slowdown in the economic growth, there is an urgent need to transform China's entire industrial chain. To promote an all-around industrial upgrading, China has proposed the strategy of ''Internet +'', and the integration of China's ecommerce with its traditional economy has been significantly improved. Ecommerce has accelerated its expansion from consumption to various industries and infiltrated all aspects of social and economic activities, thereby driving the development of enterprise-level ecommerce, both in scope and in depth, and facilitating the transformation and upgrading of enterprises. The Monitoring Report on the Data of China's Ecommerce Market [1] shows that in 2016, the volume of ecommerce transactions in China reached approximately 3.5 trillion dollars, a year-on-year growth rate of approximately 25.5%.

The rapidly rising number of cyber-transactions has spawned ecommerce big data. As increasingly numerous data files are being stored locally in enterprises, the pressure on local data storage systems greatly increases. Local hardware failures lead to great damage or loss of data, which greatly affects the daily operations of the enterprises. Fortunately, cloud storage techniques came into being under such circumstances. Cloud computing can collect and organize a large number of different types of storage devices by means of various functions, such as cluster applications, network technology and distributed file systems. There have already been a number of typical cloud service products at home and abroad, such as Amazon Web Services [2], Microsoft Azure [3], i Cloud [4], and App Engine [5].

As large amounts of data are outsourced to cloud storage servers, the need for data owners to encrypt the above-mentioned second and third types of sensitive data makes traditional plain text-based data search solutions no longer suitable. In addition, restricted by the network bandwidth and local storage capacity constraints, users find it impossible to re-download all the data to a local disk and later decrypt them for use. Based on the above issues, privacy-preserving data search schemes were born, designed to ensure that only legitimate users based on identifiers or keywords, and have the ability to search the data. These schemes safeguard the users' personal data but enable the server to return to the target ciphertext file according to the query request. Thus, we can ensure the security of user data and privacy while not unduly reducing the query efficiency.

In this paper, we focus on the second and third types of data and design a secure and efficient data search scheme. For convenience, a practical background is presented as follows. We first assume that each product has a unique identifier in

the whole company and a detailed description file. The file includes all of the detailed information of the product, such as the design flow, design standard, product features and market position. As we all know, launching the product to the market earlier than the competitor can occupy the market quickly and benefit the company considerably. As a consequence, all of the information should be kept from the competitors and the public, considering that the products are time-sensitive.

With the growth of the company, product information also increases greatly. To improve the stability and reliability of a data storage system, an intuitive scheme is moving the local data management system to the cloud. Cloud computing is widely treated as a promising information technique (IT) infrastructure because of its powerful functionalities. It can collect and reorganize huge resources of storage, computing and applications, which means that the users can access the IT services in a flexible, ubiquitous, economic and on-demand manner [10]. An accompanying challenge is how to protect the confidentiality of the data while maintaining its searchability In this paper, we design an encrypted product information retrieval system. This system includes two index structures: a hash value index tree, known as an ID-AVL tree, and a height-balanced index tree, known as a product retrieval feature (PRF) tree. Based on the two index trees, two data search methods are supported, i.e., the data users can search the desired product by the identifier or feature vector. The elements in the ID-AVL tree are the hash values of the product identifiers, rather than the plaintext data, and the tree thus can be directly outsourced to the cloud. Meanwhile, the elements in the PRF tree are plaintext data, and they are encrypted by the secure kNN algorithm before being outsourced. In addition, a detailed depth-first product search algorithm is designed for the PRF tree. Simulation results show the effectiveness and efficiency of the proposed scheme.

We summarize the primary contributions of this paper as follows:

- A product information outsourcing and searching system model including the data owner, cloud server and data users is designed.
- Two index structures supporting efficient product retrieval are constructed. Moreover, corresponding search algorithms are also proposed.
- We integrate the secure kNN algorithm into our scheme to guarantee the security of the outsourced data while maintaining the searchability.
- A series of simulations are conducted to illustrate the security and efficiency of the proposed scheme.

The rest of this paper is organized as follows. We first summarize the related work of privacy-preserving data search schemes in Section 2. Next, the data search system model and preliminary techniques are discussed in Section 3. Section 4 presents the encrypted product information retrieval scheme in detail, and the evaluation of the proposed scheme is provided in Section 5. Finally, the study's conclusions are presented in Section 6.

II. RELATED WORK

Cloud storage services have several advantages, such as ease of use and cost saving, and they are widely used in many fields. However, several challenges are associated with them. With the increasing popularity of cloud storage, security issues have become an important factor restricting its development. In recent years, data leakage accidents have repeatedly occurred in such companies as Microsoft, Google, Amazon, and China's Home Inn, Hanting, and Ctrip, and these incidents have exacerbated users' worries.

To counter the information leakage, data owners and enterprises typically outsource the encrypted business data, rather than the plaintext data, to cloud storage servers. In general, the outsourced data can be divided into three types. The first type is the open-resource-type data, which do not need to be hidden from the cloud server, such as the basic information of the enterprise and the parameters of products. The second type is the private data, which need to be encrypted but are only accessed and decrypted by the data contributor [6], [7]. This type includes such data as internal confidential information, intellectual properties and patents. The third type is the private data that need to be encrypted but can also be shared with specific users or groups [8], [9]. This type includes internal shared data, hospital's division-wide case information and information by some shared advanced users.

A single keyword Boolean search [7], [11]–[16] is the simplest document retrieval method for encrypted files. Song *et al.* [7] first proposed the searchable encryption scheme in which each word in a document is encrypted independently, and the users need to scan the entire document to search for a certain keyword. Consequently, this method has an extremely high searching complexity. Next, Goh [11] formally built the security definitions for symmetric searchable encryption, and a scheme based on a Bloom filter was designed. The security definitions are extended in [12] and [17]. Due to the lack of a rank mechanism for the returned results, the data users need to take a long time to screen the returned results, which is unacceptable in general. Thus, many single keyword-ranked search schemes have been proposed [13]–[15], [18], [8]. Though these schemes can return more accurate search results, they cannot satisfy users' requirements in most cases, considering that a single word cannot provide sufficient information to describe the users' interests.

Multiple keyword Boolean search schemes allow the data users to input a set of keywords to search the desired documents. Conjunctive keyword search schemes [19]–[21] return the documents in which all the keywords specified by the search query appear; disjunctive keyword search schemes return all the documents that contains at least one keyword of interest. Predict keyword search schemes [22]–[24] have been proposed to support both conjunctive and disjunctive search patterns. However, the returned results are still not sufficiently suitable to the users because the degrees of importance of the keywords are not considered in these schemes.

Cao *et al.* [25] first proposed a basic privacy-preserving multi-keyword ranked search scheme based on a secure kNN algorithm [26]. A set of strict privacy requirements are established, and two schemes are later proposed to improve the security and search experience. However, an apparent drawback of this scheme is that the search efficiency is linear with the cardinality of the document collection, and consequently, it cannot be used to process extremely large document databases. Xia *et al.* [27] designed a keyword balanced binary tree to organize the document vectors and proposed a "Greedy Depth-First Search" algorithm to improve the search efficiency. Moreover, the index tree can be updated dynamically with an acceptable communication burden. Chen *et al.* [28] took the relationships of documents into consideration, and a hierarchical-clustering-based index structure was designed to improve the search efficiency. In addition, a verification scheme was also integrated into their scheme to guarantee the correctness of the results. However, these two index trees in [27] and [28] can be further improved in terms of efficiency and accuracy as discussed in Section 1. Fu *et al.* [29] presented a personalized multi-keyword ranked search scheme in which an interest model of the users is integrated into the document retrieval system to support a personalized search and improve the users' search experience. Specifically, the interest model of a data user is built based on his search history with the help of WordNet [30] in order to depict his behaviors in fine grit level. However, this scheme does not support dynamic update operations because the document vectors are constructed based on all the documents. In addition, though an MDB-tree is employed to improve the search efficiency, the effectiveness of the tree is difficult to predict. Several other related studies in the field of cloud computing can be found in [33]–[37].

III. SYSTEM MODEL AND THE INDEX TREES

A. PRODUCT RETRIEVAL SYSTEM

As shown in Fig. 1, the entire product retrieval system model is composed primarily of three entities: the data manager, the cloud server and the data user. The primary responsibilities of these three entities are presented in the following.

The data manager is responsible for managing the product and collecting the product information. In addition, the data manager needs to encrypt the product information file by a symmetric encryption technique before outsourcing the data to the cloud server. To improve the security of the files, each file is encrypted by a single secret key, and the keys of different files are independent. Furthermore, to improve the search efficiency, an index structure is constructed for the outsourced data. At first, an identifier index structure is constructed based on the hash function and height-balanced binary search tree. Then, a feature vector tree is built for all the feature vectors of the product, and it is encrypted by the secure kNN algorithm.

When a data user wants to search a set of chosen products, she needs to generate a trapdoor to describe her interest. Two types of the trapdoor can be provided, i.e., a set of hash values of the desired product information files or a set of feature vectors. For the first type of trapdoor, a set of encrypted files with the same hash identifiers are returned, and for the second type trapdoor, the most relevant encrypted files are returned. The data user can obtain the plaintext files by decrypting the returned files with the help of the symmetric secret keys. These secret keys are provided by the data manager.

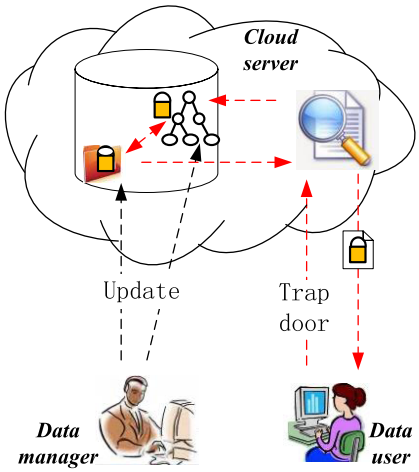

FIGURE 1. Encrypted product information retrieval system model.

The cloud server stores all the data uploaded by the data manager. When a data user needs to search the data in the cloud, she first generates a trapdoor, which is sent to the cloud server. A search engineer is employed by the cloud server to act as a bridge between the data users and the encrypted data. Though the cloud server cannot get the plaintexts of the data, it should be capable of sending the accurate search result of the trapdoor to the data users. Of course, the returned data are ciphertext, and the data user needs to decrypt them by the symmetric secret keys which are provided by the data manager.

B. ID-AVL TREE

To construct the ID-AVL tree, we first encrypt all the product identifiers based on a hash function, hash(). Next, each node in the ID-AVL tree contains a hash value of the product ID, and all of the hash values are organized based on an AVL tree [31] as shown in Fig. 2. Two important properties of AVL, which can help us to maintain the hash values, are presented as follows. First, the ID-AVL tree can be updated flexibly by inserting a node, deleting a node and modifying a node. Correspondingly, we can update the ID-AVL tree from time to time by changing the product information. Second, the values of the left child nodes of a parent node are always smaller than that of the parent node; the values of the right child nodes of a parent node are always larger than that of the parent node. In theory, the time complexity of inserting, deleting and searching a node are all log(N), where N is the number of nodes in the tree. In this paper, we construct the ID-AVL tree based on the algorithm in [31].

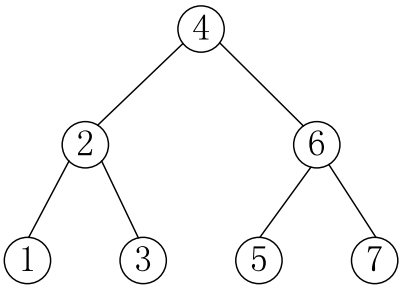

FIGURE 2. Product hash value index tree.

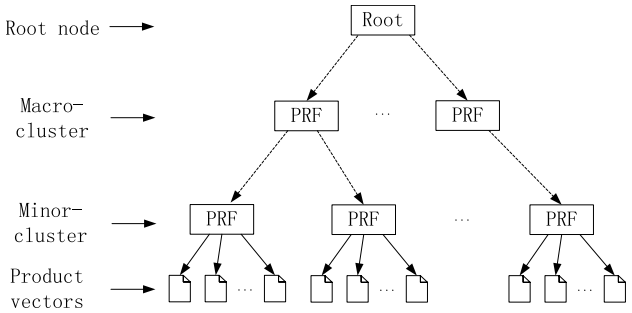

FIGURE 3. Product retrieval feature tree.

C. PRODUCT RETRIEVAL TREE

The feature dictionary of the products is denoted as $D = \{f_1, f_2, \cdots, f_m\}$, and the feature set S_i of any product P_i must be a subset of D, i.e., $S_i \in 2^{\{f_1, f_2, \cdots f_m\}}$. Then, the feature vector V_i of product P_i is constructed as follows.

- The initial vector of P_i is a $1 \times m$ vector, and all the elements in the vector are 0;
- We orderly scan all the elements in the initial vector and assign a value to the element of feature f_i if the feature of P_i can be quantized.
- Based on the different degrees of importance of the features, a weight is employed to multiply the elements in the vector to reflect this.

To search the product information, a trapdoor needs to be constructed by the data user in a similar way, and the similarities between the trapdoor V_Q and the product feature V_i is calculated as $\text{sim}(V_i, V_Q) = V_i \cdot V_Q$. Moreover, the similarity between two the vectors V_i and V_j is defined as $\text{sim}(V_i, V_j) = V_i \cdot V_j$. Next, the product feature vectors are organized as hierarchical clusters according to their similarities. Each node in the tree represents a cluster composed of a set of product feature vectors or sub-clusters. The PRF vector of a node is a quintuple summarization about a cluster.

Given K m-dimensional product feature vectors in a cluster: $\{V_j\}$ where $j = 1, 2, \cdots, K$, the PRF vector of the cluster is denoted as a quintuple: $PRF = (K, LS, SS, V_{min}, V_{max})$, where K is the number of product feature vectors in the cluster, LS is the linear sum of the K product feature vectors, i.e., $LS = \sum_{j=1}^{K} V_j$, SS is the square sum of the K product feature vectors, i.e., $SS = \sum_{j=1}^{K} V_j^2$ (SS is a numerical value rather than a vector), V_{min} denotes a vector consisting of m values which are calculated as follows:

$$V_{min}[i] = min(V_1[i], V_2[i], \cdots, V_K[i]), \tag{1}$$

where $V_j[i]$ is the i-th dimensional value of V_j, and similarly, V_{max} is calculated as follows:

$$V_{max}[i] = max(V_1[i], V_2[i], \cdots, V_K[i]). \tag{2}$$

Based on a PRF vector, the centroid of a cluster C can be easily calculated as

$$c = LS/K, \tag{3}$$

and the relevance score between cluster C and a product vector V_j is defined as

$$\text{RScore}(C, V_j) = c \cdot V_j. \tag{4}$$

Similarly, the relevance score between cluster C and a query vector V_Q is defined as

$$\text{RScore}(C, V_Q) = c \cdot V_Q. \tag{5}$$

Further, the radius of cluster C is defined as follows:

$$R = \sqrt{\sum_{j=1}^{K} (V_j - c)^2 \Big/ K}, \tag{6}$$

and it can be calculated by the PRF vector as follows:

$$R = \sqrt{(SS - LS^2/K)/K}. \tag{7}$$

Theorem 1 (PRF Additivity Theorem): If we merge two disjoint clusters with PRF vectors: $PRF_1 = (K_1, LS_1, SS_1, V_{min1}, V_{max1})$ *and* $PRF_2 = (K_2, LS_2, SS_2, V_{min2}, V_{max2})$*, then the PRF vector of the combined cluster is*

$$\begin{aligned} PRF &= PRF_1 + PRF_2 \\ &= (K_1 + K_2, LS_1 + LS_2, SS_1 + SS_2, V_{min}, V_{max}), \end{aligned} \tag{8}$$

where $V_{min}[i] = \min(V_{min1}[i], V_{min2}[i])$ *and* $V_{max}[i] = max(V_{max1}[i], V_{max2}[i])$.

Proof: The proof consists of straightforward algebra. □

In the similar way, we can obtain the PRF Subtraction Theorem, which can be used to divide two clusters, though V_{min} and V_{max} need to be recalculated. The structure of a PRF tree is presented in Fig. 3. It can be observed that each leaf node is composed of a set of similar product vectors and its PRF vector is directly extracted from the product vectors. The similar leaf nodes agglomerate with each other to compose the non-leaf nodes until all the product vectors belong to a huge cluster at the a root node. Based on Theorem 1, the PRF vectors of the non-leaf nodes and the root node are calculated based on the PRF vectors of all their child nodes.

IV. ENCRYPTED PRODUCT INFORMATION RETRIEVAL SCHEME

A. CONSTRUCTION OF PRODUCT RETRIEVAL TREE

A PRF tree has three main parameters: branching factors B_1, *and* B_2 and threshold T, which are preset by the data

owner. Each non-leaf node NL_i contains at most B_1 child nodes, and it is defined as follows:

$$NL_i = (PRF, PRF_1, child_1, \cdots, PRF_{B_1}, child_{B_1}) \quad (9)$$

where PRF is the PRF vector of the whole cluster, PRF_i is the PRF vector of the i-th sub-cluster and $child_i$ is a pointer to the child node representing the sub-cluster. A non-leaf node represents a cluster made up of all the sub-clusters represented by its child nodes. A leaf node L_i contains at most B_2 product vectors, and it is defined as follows:

$$L_i = (PRF, child_1, \cdots, child_{B_2}), \quad (10)$$

where PRF is the PRF vector of the cluster, $child_i$ is a pointer to the i-th product vector in the cluster. Furthermore, the cluster of a leaf node must satisfy a threshold requirement: the radius of the cluster (11) must be less than T. The default values in the nodes are set to *null*.

The PRF tree is constructed in an incremental manner, and the process of inserting a product vector V_j into the PRF tree is presented as follows:

- *Identifying the appropriate leaf node*: Starting from the root, V_j recursively descends the PRF tree by choosing the closest child node according to the relevance scores between V_j and the sub-clusters as defined in (11) until it reaches a leaf node.
- Modifying the leaf node: When V_j reaches a leaf node L_i, it tests whether L_i can "absorb" V_j without violating the constraints of B_2 and T. If so, V_j is inserted into L_i and the PRF vector of L_i is updated. If not, we must split L_i to two leaf nodes. Node splitting is performed by choosing the farthest pair of product vectors as seeds and redistributing the remaining product vectors based on the closest criteria. The PRF vectors of the two new leaf nodes need to be recalculated.
- Modifying the path from the root node to the leaf node: After inserting V_j into a leaf node, we need to update the PRF vector for all the nodes on the path to the leaf node. In the absence of a split, this simply involves updating PRF vectors based on Theorem 1. A leaf node split requires us to insert a new leaf node into the parent node. If the parent node has space for the new leaf node, we only need to insert the new leaf node into it and then update the PRF vector for the parent node. In general, however, we may have to split the parent node as well, and so on, up to the root. If the root is split, the tree height increases by one.

B. RETRIEVAL PROCESS OF THE INTERESTED PRODUCTS

In this paper, the data users can retrieve the interested product in two ways, i.e., retrieving the products by their identifiers or the product feature vector. When a data user wants to search a product based on its identifier, she first needs to encrypt the identifier based the on the hash function, hash(). Next, the hash value of the identifier is sent to the cloud server. The cloud server is responsible for searching for the hash value in the ID-AVL tree, and once the hash value is found, the corresponding encrypted production information is sent to the data user. Finally, the data user can decrypt the product information based on the secret keys, and the data retrieval process is completed.

Moreover, in certain cases, the data user may want to search the product based on the features. Initially, the data user needs to construct the feature vector of the product as discussed in Section 3.3. Then, we need to design a depth-first search algorithm for the PRF tree, and that algorithm is presented in Algorithm 1.

Algorithm 1 DepthFirstSearch(a PRF Tree With Root r, a Query Vector V_Q)

1: $u \leftarrow r$;
2: **while** u is not a leaf node
3: Calculate all the relevance scores between the child nodes of u with V_Q based on (5);
4: $u \leftarrow$ the most relevant child node;
5: **end while**
6: Select the most relevant k document vectors in u by RScore(V_i, V_Q) and construct *RList*;
7: *Stack*.push(r);
8: **while** *Stack* is not empty
9: $u \leftarrow$*Stack*.pop();
10: **if** the node u is not a leaf node
11: **if** RScore $\left(V_{u,max}, V_Q\right) > k$thScore
12: Sort the child nodes of u in ascending order based on the relevant scores with V_Q;
13: Push the children of u into *Stack* in order, i.e., the most relevant child is latest inserted into *Stack*;
14: **else**
15: **break;**
16: **end if**
17: **else**
18: Calculate the relevance scores between the document vectors in the leaf node with V_Q and update *RList*;
19: **end if**
20: **end while**
21: **return** *RList*;

In Algorithm 1, the $k$thScore represents the smallest relevance score in the current result list *RList*, which stores the most k relevant accessed document vectors with V_Q and the corresponding relevance scores. In addition, we employ the variable *Stack* to store the nodes which need to be searched in the future. In addition, *Stack*.push(u) inserts node u into *Stack* and *Stack*.pop() returns the latest inserted node. In the initial phase, we need to first locate the most relevant leaf node with the query vector in the tree to initialize *RList* as presented in line 2 to line 6. Then, the result list is continuously updated by searching the necessary paths in the tree until the final search result is obtained as presented in line 8 to line 19. Compared with the search process of the keyword balanced binary tree proposed in [27], the search process presented in

Algorithm 1 is much more efficient considering that many search paths are pruned in the searching process.

C. ENCRYPTION OF THE PRODUCT RETRIEVAL TREE

For each product P_i, two types of information are first extracted, including its identifier i and the product vector V_i. We encrypt the identifier i through a hash function, hash(). The construction process of the $ID-AVL$ tree is presented as follows. The constructed $ID-AVL$ tree can be directly outsourced to the cloud server because it stores only a set of hash values, rather than the plaintext identifier.

Based on the product vectors, the process of building the PRF tree has been presented in Section 4.2. In contrast to the $ID-AVL$, the PRF tree needs to be encrypted before being outsourced. In the PRF tree, we treat LS, V_{min} and V_{max} to the same as product vectors and encrypt them in the same way. Note that parameter K in a PRF vector does not need to be encrypted, and SS, which will not be used in the search process, does not need to be sent to the cloud server. Before encrypting a product vector V_j in the PRF tree, we first extend it to $(m+m')$ dimensions. In addition, we spilt each dimension of $V_j[i]$ into $V_j[i]'$ and $V_j[i]''$. Specifically, if $S_{2i}=0$, $V_j[i]'$ and $V_j[i]''$ will be set equal to $\mathrm{V_j[i]}$; otherwise, $\mathrm{V_j[i]}'$ and $\mathrm{V_j[i]}''$ will be set as two random numbers whose sum is equal to $\mathrm{V_j[i]}$. Next, we randomly select two invertible matrices $\mathrm{M_1}$, $\mathrm{M_2}$ and encrypt V_j as $E_j=\{M_1^T V_j', M_2^T V_j''\}$.

Once a search request $\mathcal{SR}$ is received by the proxy server, it first extracts its parameters including ID' and $v_{\mathcal{SR}}$. Parameter ID' is encrypted by hash() and we get $h_{ID'}$. We extend $v_{\mathcal{SR}}$ to $(m+m')$ dimensions. Specifically, if $S_{1i}=0$, the i-th dimension of V_Q corresponds to a feature w_r, which is extracted from W in order, and $V_Q[i]$ is set to w_{w_r}; otherwise, this dimension is an artificial dimension and $V_Q[i]$ is set to a random number. Note that the value of the last artificial dimension is not a random number, and it should be calculated carefully to guarantee that the dot product of the artificially added dimensions in the product vectors and in V_Q is 0. Further, we spilt $V_Q[i]$ into $V_Q[i]'$ and $V_Q[i]''$. Specifically, if $S_{2i}=1$, $V_Q[i]'$ and $V_Q[i]''$ will be set equal to $V_Q[i]$; otherwise, $V_Q[i]'$ and $V_Q[i]''$ will be set as two random numbers whose sum is equal to $V_Q[i]$. Finally, we encrypt V_Q as $E_Q=\{M_1^{-1}V_Q', M_2^{-1}V_Q''\}$. In this case, the relevance score of V_j and V_Q defined in Section 3.2 can be calculated as follows:

$$\mathrm{RScore}\left(V_j, V_Q\right) = V_j \cdot V_Q = E_j \cdot E_Q. \tag{11}$$

The trapdoor $\mathcal{TD}$ is composed of the hash values of the filename and authors and E_Q.

V. PERFORMANCE EVALUATION

A. SECURITY ANALYSIS

In our scheme, the outsourced data includes the product information file, ID-AVL tree and PRF tree. The product information files are encrypted symmetrically based on the independent secret keys, and the cloud server does not have the secret keys. In this case, the plaintext files cannot be decrypted by the cloud server. In the ID-AVL tree, the stored values are the hash values of the product identifiers, and they contain no valuable information about products. The PRF tree is encrypted by the secure kNN algorithm before being outsourced to the cloud server. Though the cloud server knows the encrypted feature vectors in the tree, the cloud server does not know the matrices M_1, M_2; hence, the plaintext vectors in the tree cannot be recovered.

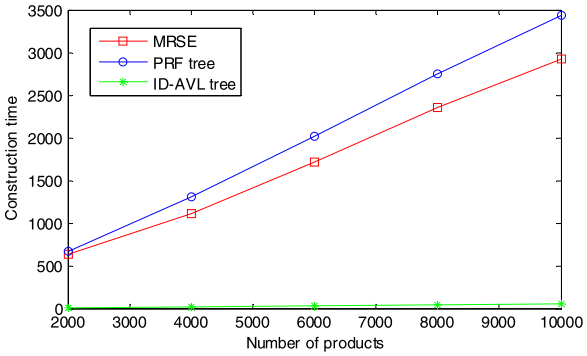

FIGURE 4. Construction time of the index structures.

B. PRODUCT INFORMATION SEARCH EFFICIENCY

In this section, we evaluate the search efficiency of our scheme. First, we evaluate the construction time of the index structures of the product information. Specifically, we compare our scheme with the MRSE scheme [25]. To decrease the bias of the data manager who is responsible for generating the vectors and the hash values, in this paper we employ the Enron Email Data Set [32] to test our scheme. Specifically, the data set is employed to act as the product information files. Moreover, the vectors of the product are assumed to be extracted from the data set based on the TF-IDF model, and then the vectors are organized by the PRF tree.

As shown in Fig. 4, with the increasing number of products, the construction time of PRF tree and the index structures in MRSE monotonously increase. This is reasonable considering that each product information file needs to be scanned for a time to get the feature vectors. The construction time of the PRF tree is slightly longer than that of the MRSE scheme, because the vectors need to be further inserted to the trees in the PRF tree. Apparently, the ID-AVL tree is considerably simpler, and the construction time can be ignored compared with the other two trees.

To search the desired product information, the data user needs to first generate the trapdoor, which is sent to the cloud server. The times of constructing the trapdoors with the increasing of the size of the feature dictionary are presented in Fig. 5. The search requests based on the identifiers are independent of the feature dictionary, and hence, the time of constructing the trapdoors for the ID-AVL tree remains stable. However, the construction time of the trapdoors for the MRSE and PRF trees monotonously increase with the

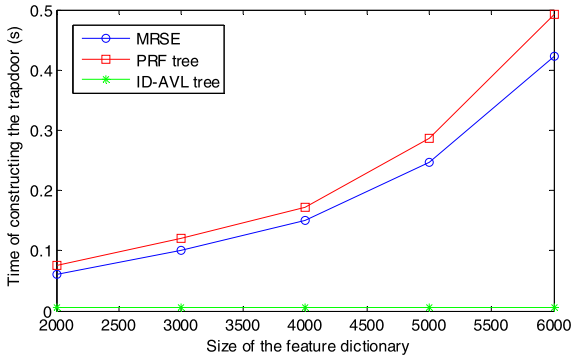

FIGURE 5. Time of constructing the trapdoors.

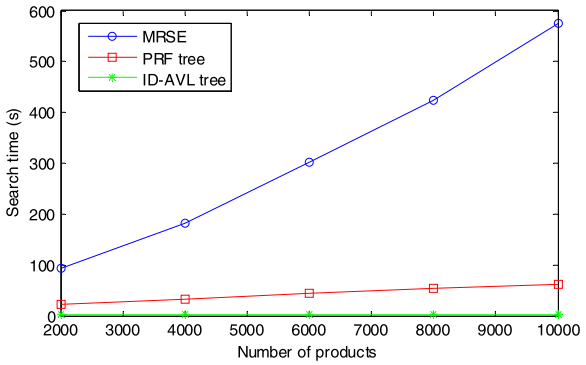

FIGURE 6. Search time with different number of products.

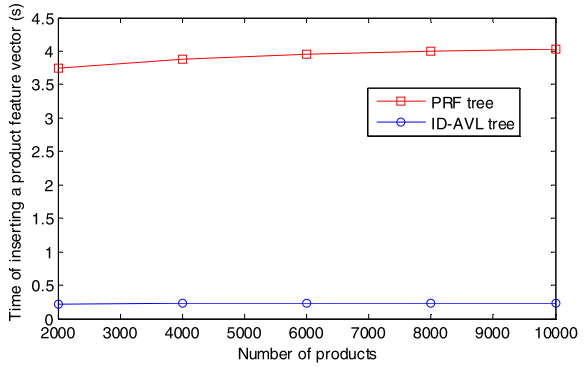

FIGURE 7. Time consumption of inserting a node into the trees.

increasing of the feature dictionary's size. This is reasonable considering that the size of the product feature vector is equal to the size of the feature dictionary. In addition, the time costs for the MRSE and PRF trees are similar to each other because the processes of generating the trapdoors are similar.

The search time of a trapdoor in the cloud server is presented in Fig. 6. It can be observed that the MRSE scheme consumes the most time to execute a search operation. Moreover, the search time increases monotonously with the increasing of the number of products. This increase can be explained by the fact that in MRSE, the feature vectors are stored in order, and they do not employ any index structure. In this case, the cloud server needs to scan all the product feature vectors to get the search result. The PRF tree organizes the vectors by a height-balanced tree, and most paths in the tree are pruned in the search process. As a consequence, the search efficiency is greatly improved. Finally, we can observe that the ID-AVL tree is the most efficient index structure, which can be explained by the fact that the ID-AVL tree is considerably simpler, and the search process is also very easy.

With the expanding growth of companies, more and more product information needs to be outsourced to the cloud server. Consequently, we need to update the index trees from time to time, and the update efficiency also affects the performance of our scheme significantly. As shown in Fig. 7, the update time of both the PRF tree and the ID-AVL tree increases slightly with the increasing number of products, which is reasonable, considering that we need to search the trees to identify the proper location of the inserted node. In addition, updating the PRF tree consumes much more energy than that of updating the ID-AVL tree. This can be explained by the fact that the ID-AVL tree is much simpler than the PRF tree, and in theory only $\log(N)$ nodes need to be searched. Though quite many paths in the PRF tree are pruned in the search process, the number of the search paths is considerably larger than $\log(N)$ and more time is thus consumed in the PRF tree.

VI. CONCLUSIONS

In this paper, we designed a secure and efficient product information retrieval scheme based on cloud computing. Specifically, two index structures, including a hash value AVL tree and a product vector retrieval tree, are constructed, and they support an identifier-based product search and feature-vector-based product search, respectively. Correspondingly, two search algorithms are designed to search the two trees. To protect the product information privacy, all the outsourced data are encrypted. The product information is symmetrically encrypted based on a set of independent secret keys, and the product vectors are encrypted based on the secure kNN algorithm. Security analysis and simulation results illustrate the security and efficiency of the proposed scheme.

As the future work, we attempt to seamlessly integrate more index structures into our scheme to support more search patterns. Another difficult and promising challenge is further improving the search efficiency.

REFERENCES

[1] (May 24, 2017). *2016 Monitoring Report on the Data of China's E-Commerce Market [EB/OL]*. [Online]. Available: http://www.100ec.cn/zt/16jcbg/

[2] Amazon. *Amazon S3*. Accessed: Sep. 5, 2017. [Online]. Available: http://aws.amazon.com/s3/

[3] *Windows Azure*. Accessed: Sep. 5, 2017. [Online]. Available: http://www.microsoft.com/windowsazure/

[4] *Apple i Cloud*. Accessed: Sep. 5, 2017. [Online]. Available: http://www.icloud.com/

[5] *Google App Engine*. Accessed: Sep. 5, 2017. [Online]. Available: http://appengine.google.com/

[6] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, 2004, pp. 31–45.
[7] D. X. Song and D. A. Wanger Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, May 2000, pp. 44–55.
[8] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, pp. 506–522.
[9] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *J. Syst. Softw.*, vol. 83, no. 5, pp. 763–771, 2010.
[10] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.
[11] E.-J. Goh, "Secure indexes," IACR Cryptol. ePrint Arch., Newark, NJ, USA, Tech. Rep. 1, 2003, p. 216.
[12] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, 2011.
[13] A. Swaminathan *et al.*, "Confidentiality-preserving rank-ordered search," in *Proc. ACM Workshop Storage Secur. Survivability*, 2007, pp. 7–12.
[14] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.
[15] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber^{+R}: Top-k retrieval from a confidential index," in *Proc. 12th Int. Conf. Extending Database Technol., Adv. Database Technol.*, 2009, pp. 439–449.
[16] S. Jarecki, C. Jutla, H. Krawczyk, M. Rosu, and M. Steiner, "Outsourced symmetric private information retrieval," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 875–888.
[17] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Network Secur.*, 2005, pp. 442–455.
[18] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. IEEE 30th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2010, pp. 253–262.
[19] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. Int. Conf. Inf. Commun. Secur.*, 2005, pp. 414–426.
[20] Y. Ho Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. Int. Conf. Pairing-Based Cryptogr.*, 2007, pp. 2–22.
[21] B. Zhang and F. Zhang, "An efficient public key encryption with conjunctive-subset keywords search," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 262–267, 2011.
[22] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2010, pp. 62–91.
[23] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. Theory Cryptogr. Conf.*, 2009, pp. 457–473.
[24] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2008, pp. 146–162.
[25] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[26] W. K. Wong, D. W. L. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.
[27] Z. Xiam, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.
[28] C. Chen *et al.*, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.
[29] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.
[30] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM* vol. 38, no. 11, pp. 39–41, 1995.
[31] G. Adelson-Velsky and G. E. Landis, "An algorithm for the organization of information," (in Russian), in *Proc. USSR Acad. Sci.*, vol. 146. 1962, pp. 263–266.
[32] W. W. Cohen. (2015). *Enron Email Data Set*. [Online]. Available: https://www.cs.cmu.edu/~./enron/
[33] C. Zhu, J. J. P. C. Rodrigues, V. C. M. Leung, L. Shu, and L. T. Yang, "Trust-based communication for the industrial Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 16–22, Feb. 2018.
[34] C. Zhu, L. Shu, V. C. M. Leung, S. Guo, Y. Zhang, and L. T. Yang, "Secure multimedia big data in trust-assisted sensor-cloud for smart city," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 24–30, Dec. 2017.
[35] C. Zhu, H. Zhou, V. C. M. Leung, K. Wang, Y. Zhang, and L. T. Yang, "Toward big data in green city," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 14–18, Nov. 2017.
[36] C. Zhu, V. C. M. Leung, K. Wang, L. T. Yang, and Y. Zhang, "Multi-method data delivery for green sensor-cloud," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 176–182, May 2017.
[37] C. Zhu, X. Li, V. C. M. Leung, L. T. Yang, E. C.-H. Ngai, and L. Shu, "Towards pricing for sensor-cloud," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 74–86, Jan. 2018.

YING-SI ZHAO was born in Beijing, China, in 1984. She received the B.S. and M.S. degrees in communication engineering and the Ph.D. degree in management from Beijing Jiaotong University, Beijing, in 2007, 2009, and 2014, respectively.

From 2007 to 2010, she was a Sales Engineer of Motorola Inc. Since 2014, she has been a Teacher with the Business Administration Department, School of Economics and Management, Beijing Jiaotong University. She has authored over 10 articles, and she has presided over seven projects as a Principal Investigator. Her research interests mainly include marketing, innovation and entrepreneurship, cloud computing and its applications, network public opinion, and so on.

QING-AN ZENG (SM'02) received the Ph.D. degree in electronic engineering from Shizuoka University, Japan, in 1997. He is currently a Faculty Member with the Department of Computer Systems Technology and the Director of the Wireless and Mobile Networking Laboratory, North Carolina A&T State University, USA. He has published over 100 books, book chapters, refereed journal papers, and conference proceeding papers. His research interests are in all areas of wireless and mobile networks, ad hoc and sensor networks, handoff, mobility management, heterogeneous networks, system modeling and performance analysis, simulations, QoS, security, NoC, smart grid, smart grid communications, PLC, social networks, and queuing theory.