Received February 11, 2018, accepted March 25, 2018, date of publication April 6, 2018, date of current version July 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2823718

Privacy-Preserving and Dynamic Multi-Attribute Conjunctive Keyword Search Over Encrypted Cloud Data

LILI ZHANG[1,2], YUQING ZHANG[1,3,4], AND HUA MA[1,4]

[1]National Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
[2]Information Engineering College, Henan University of Science and Technology, Luoyang 471023, China
[3]National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences, Beijing 100049, China
[4]State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China

Corresponding author: Yuqing Zhang (zhangyq@ucas.ac.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800700, in part by the National Natural Science Foundation of China under Grant 61272481 and Grant 61572460, in part by the National Information Security Special Projects of the National Development and Reform Commission of China under Grant (2012)1424, and in part by the Open Project Program of the State Key Laboratory of Information Security under Grant 2017-ZD-01 and Grant 2016-MS-02.

ABSTRACT With the increasing popularity of cloud computing, a growing data owners are motivated to outsource their huge data to cloud servers in order to facilitate access and save data management cost. To protect user privacy and data security, sensitive data should be encrypted before outsourced to the cloud server, which obsoletes data utilization like efficient search over encrypted data. In this paper, we present a privacy-preserving conjunctive keyword search scheme over encrypted cloud data, which simultaneously supports dynamic update operations. Specifically, we construct an index structure based on multi-attribute tree (MAT) and present an efficient search algorithm over the index tree, named as the searchMAT algorithm. We propose a multi-attribute conjunctive keyword search scheme based on MAT, named as the MCKS-MAT scheme, which can achieve equality conjunction, subset conjunction and range conjunction, as well as satisfy privacy requirements under the known background attack model. In addition, this paper is accompanied by an adequate of experiments for evaluating the effectiveness of the proposed scheme. Experiments demonstrate that, compared to the linear search, the proposed scheme needs the slightly higher preprocessing cost on account of constructing the tree-based index, however, it achieves lower computational overhead in initialization, trapdoor generation and queries.

INDEX TERMS Conjunctive keyword search, cloud computing, multiple attribute tree, privacy-preserving search, tree-based index.

I. INTRODUCTION

Cloud computing [1] has become a new computing paradigm, and it can provide huge resources of storage and computing and enable users to enjoy ubiquitous and convenient network access to a great many shared computing resources efficiently with minimal economic cost. Attracted by these appealing features, many individuals and companies are motivated to put their data in the cloud, rather than purchasing software or hardware to manage their data. Despite the tremendous powerful advantages, privacy concern is one of the primary obstacles which prevent the widespread adoption of cloud computing by potential users. The main reason is that they are afraid of losing control of their data when their sensitive data are outsourced to the remote cloud server.

To protect data security and user privacy, data are usually encrypted before data outsourcing and stored in the form of ciphertext in the cloud server. Although encryption can protect data privacy against the unauthorized entities, it also activates the inconvenience of using encrypted data for the authorized user. One fundamental and common form of data utilization is the keyword search operation, i.e., to quickly sort out matching information from the huge amount of data sets according to specific search keywords. Many existing keyword-based search techniques, which are widely adopted on the plaintext data, cannot be directly used on the ciphertext data. In order to solve this problem, some general-purpose solutions are proposed based on fully-homomorphic encryption [2] or oblivious RAMs. However, these methods are impractical because of their high computational overhead. On the contrary, more practical solutions, such as Searchable Encryption (SE) [3], [4], have made specific contributions to addressing the need for secure search over outsourced data. SE is a cryptographic primitive that supports keywords search over encrypted data, which not only saves huge network bandwidth and computation resources for users but also migrates the burdensome search operation to the cloud server to utilize its strong computational capacity. So far, researchers have developed abundant SE schemes to achieve various search functionality, such as single keyword search, multi-keyword conjunctive search, ranked search, similarity search, etc. Among them, multi-keyword conjunctive search obtains more and more attention in many scenarios, such as e-mail, electrical medical records (EMRs), social network profiles. Take an example, with the rapid growth of the patients' number, a lot of hospitals or healthcare organizations outsource their encrypted EMRs to the remote server to facilitate the storage and management of their EMRs. Each record has various fixed attributes, which generally refer to a class of properties of a patient, such as "age", "gender", and "illness". The problem of retrieving all the records satisfying a query condition containing a multiplicity of attributes, known as conjunctive keyword search, is a major concern for the healthcare organization. For example, an authorized user retrieves all the records satisfying a query "($60 \leq$ age ≤ 70) and (illness = hepatitis)" with the purpose of statistical analysis. Existing simple conjunctive keyword search schemes [5]–[7] support only multi-attribute conjunctive equality queries such as a query "(age = 60) and (illness = diabetes)". In order to enrich search functionality, complex multi-attribute conjunctive keyword search schemes [8]–[10] supporting range and subset queries, have been proposed. In these schemes, a general way is to construct vectors in complex algebraic structure, such as composite order group or dual pairing vector space (DPVS). Nevertheless, the efficiency is of great concern since computation overhead of bilinear pairing on the complex algebraic structure is much more than that on prime order group. Zhang *et al.* [11] constructed an efficient multi-attribute conjunctive keyword search scheme by shortening the length of attribute vectors, which were represented by hierarchical attribute representation method. However, the common problem with these schemes is that they only achieve linear search time, which is impractical in big data scenario. Sun *et al.* [12], [13] constructed a searchable tree-based index structure and adopted cosine measure together with TF$\times$IDF. Their scheme achieved better-than-linear search efficiency; however, it was only suitable for text search.

In practice, the data owner may need to update their data after he outsources these data to the cloud server. Thus, SE schemes are supposed to support the update of the documents. However, so far, few of existing SE schemes can achieve efficient multi-attribute conjunctive keyword search and support update operation. Kamara *et al.* [14] constructed an encrypted inverted index, achieving dynamic operation of data collection efficiently. However, their scheme was very complex to implement. Soon afterward, an improved dynamic search scheme [15] was proposed based on a keyword red-black (KRB) tree data construction. However, both of them were designed only for single keyword boolean search. Xia *et al.* [16] proposed efficient multi-keyword ranked search schemes, which supported the deletion and insertion of documents flexibly. Due to the adoption of a tree-based index structure, this scheme can achieve sub-linear search time. However, Xia's schemes were only suitable for multi-keyword text search, in which the index was constructed based on term frequency. Moreover, their schemes were unable to deal with multi-attribute subset conjunction and range conjunction search over record collection (e.g. EMRs), which has multiple attributes.

Therefore, on the premise of ensuring data security and patients' privacy, developing efficient multi-attribute conjunctive keyword search schemes over encrypted EMRs, which support update operation, becomes an especially requested issue. Towards this problem, our contributions are summarized as follows:

1) A MAT-based index structure is constructed and an efficient search algorithm over the index tree is proposed.
2) The MCKS-MAT scheme is proposed, which can support efficient multi-attribute conjunctive keyword search and flexible dynamic operation on the document collection. As a result of strategically adjusting the keyword position in the index tree and appropriately appending specific bits to the vectors corresponding to each level of the index tree, this scheme satisfies privacy requirements in the known background attack model.
3) A thorough security analysis and performance simulation of the proposed scheme is made. As a result of the special structure of the MAT-based index, the proposed scheme achieves better-than-linear search efficiency. The extensive experimental evaluation shows that our scheme is more practical than linear search except for the cost of preprocessing and constructing the encrypted index, both of which are the one-time operation.

II. RELATED WORK

Searchable encryption schemes enable the clients to store their encrypted data in the cloud and carry out keyword search over ciphertext data. Abundant SE schemes have been proposed under different threat models to achieve various search functionality. According to different cryptography primitives, SE schemes can be constructed based on public key cryptography (PKC) or symmetric key cryptography (SKC). According to the search functionality, SE mechanism mainly falls into the following three categories: 1) Single keyword Search; 2) Multi-keyword boolean search; 3) Multi-keyword rank search. The second and third categories belong to multi-keyword search.

A. SINGLE KEYWORD SEARCH

For the first time, Song *et al.* [17] proposed SKC-based single keyword SE schemes, and the search time of their scheme was $O(n)$, where n is the size of the data collection. Curtmola *et al.* [18] proposed two improved schemes, which achieved optimal search time. Later, some schemes [19]–[21] achieved single keyword rank search using order-preserving techniques. Boneh *et al.* [22] presented the first PKC-based SE mechanism, where anyone with the public key can write to the data stored on the server, but only authorized users with private key could search. This paper provides guidance to realize the more diverse SE schemes by PKC for the later researchers. A limitation common to the above schemes is that they only allow the server to identify the subset of documents that match a certain keyword but do not allow for a boolean combination of such queries.

B. MULTI-KEYWORD BOOLEAN SEARCH

Multi-keyword boolean search allows the users to input multiple query keywords to request suitable documents. Among multi-keyword boolean search schemes, simple conjunctive keyword search (a.k.a equality conjunction search) schemes only return the documents that contain all of the query keywords. Golle *et al.* [5] realized the first conjunctive keyword search based on SKC. Later, Ballard *et al.* [23] proposed two conjunctive keyword search constructions which minimized the trapdoor size and the computation overhead. Park *et al.* [24] and Hwang and Lee [25] introduced the PKC-based constructions to achieve conjunctive keyword search. Liu *et al.* [26] proposed conjunctive keyword search scheme over multiple resource data. These schemes only supported multi-keyword equality conjunction search. Boneh and Waters [9] extended conjunctive keyword functionality and constructed searchable public-key systems which supported equality conjunction, subset conjunction, and range conjunction. Shen *et al.* [7] designed a multi-dimensional range query over Encrypted Data (MRQED) scheme, in which every dimension supported range query. These two schemes were based on bilinear pairing and needed large computation overhead. Based on asymmetric scalar-product preserving encryption (ASPE), Zhang *et al.* [11] constructed efficient multi-keyword conjunctive search schemes, which supported equality conjunction, subset conjunction and range conjunction search. Disjunctive keyword search schemes return all of the documents that contain a subset of the query keywords. In recent years predicate encryption schemes are proposed to support more diverse search functionality, such as disjunctive search. Katz *et al.* [27] proposed predicate encryption supporting disjunctions and polynomial equations and inner products. The main idea of this scheme was to construct vectors in composite-order groups to represent conjunctive normal formulation and disjunctive normal formulation and polynomial equations. However, the computational overhead of the bilinear pairing constructed in the composite-order group is extremely high (about 50 times the prime order group). Based on hierarchical predicate encryption (HPE) [28], Li *et al.* [8] achieved authorized private keyword searches on encrypted personal health record, supporting range conjunction and subset conjunction query. HPE is based on DPVS which is vector space generated by n prime order groups. Bilinear pairing computation in DPVS is equivalent to the computation in n prime order group.

C. MULTI-KEYWORD RANK SEARCH

Ranked search, known as top-k search, can send back only the top-k most relevant documents. For the first time, Cao et al. [29] proposed privacy-preserving multi-keyword ranked search scheme, in which documents and queries were represented as vectors of dictionary size. Since the importance of the different keywords wasn't considered, this scheme was not accurate enough. Moreover, the search complexity was linear with the size of the data collection. To improve search efficiency and rank accuracy, Sun *et al.* [12], [13] constructed a searchable tree-based index structure and adopted cosine measure together with TF$\times$IDF. Their scheme achieved better-than-linear search efficiency. Zhang *et al.* [30] proposed a multi-keyword ranked search scheme in a multi-owner model, in which an ''Additive Order Preserving Function'' was adopted to retrieve the most relevant search results.

In practice, the data owner may need to insert or delete the documents after he uploads the document collection to the cloud server. Thus, the SE schemes are expected to support the update operation of the documents. So far, several dynamic SE schemes have been proposed. Kamara *et al.* [14] constructed an encrypted inverted index, which could achieve dynamic operation of data collection efficiently. However, their scheme was very complex to implement. Soon afterward, Kamara and Papamanthou [15] proposed an improved search scheme based on KRB tree data construction, which could achieve dynamic update on document data stored in leaf nodes. However, both of these two schemes only support single keyword boolean search. Xia *et al.* [16] proposed efficient multi-keyword ranked search schemes, which support the deletion and insertion of documents flexibly. Due to the adoption of a special tree-based index structure, their scheme can achieve sub-linear search time. However, the schemes are only suitable for text search because the index construction is based on term frequency.

III. PROBLEM FORMULATION

A. SYSTEM ARCHITECTURE AND ATTACK MODELS

The system model considered in this paper involves three entities: the data owner, the cloud server, data users, as shown in Fig. 1. In the following sections, we take outsourcing of EMRs as an example to elaborate the relevant problems.

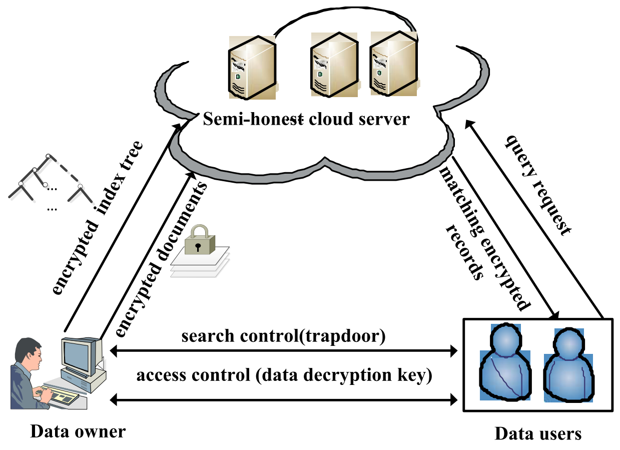

FIGURE 1. System architecture.

The data owner refers to a special type of user, i.e., a hospital or medical organization. The data owner outsources encrypted documents set $D = \{d_1, \ldots, d_n\}$, along with encrypted searchable index tree I generated from $R = \{r_1, \ldots, r_n\}$, to the cloud server. Moreover, the data owner is partially responsible for the update operation of his documents. When updating, the data owner generates update information locally and uploads it to the cloud server.

Data users are generally those who can perform the search over the encrypted data. With k query keywords, the authorized user can generate a trapdoor according to search control mechanism. Upon receiving the matching encrypted documents, the data user can decrypt the documents with the shared secret key. We assume that the data user has mutual authentication capacity with the data owner. As such, the search control mechanism, e.g., broadcast encryption can be adopted here.

The cloud server stores the encrypted document collection and encrypted index tree. When receiving the trapdoor for search, the cloud server performs matching search according to specific query rules for the user. What needs illustration is that in this paper we mainly focus on the problem of multi-attribute conjunctive search over the encrypted cloud data. Specifically, we mainly focus on the construction and encryption of the index tree and search over the encrypted index tree. The problem about encryption of the outsourced documents and decryption of the received documents is a separate issue and is out of the scope of this paper.

In the proposed scheme, the cloud server is considered "semi-honest", which is employed by a lot of related work [5], [11], [29]. Depending on the available information to the cloud server, three attack models are considered.

1) ONLY CIPHERTEXT ATTACK MODEL

In this attack model, only encrypted document set and encrypted index tree, as well as the trapdoor submitted by the data user, are known to the cloud server.

2) KNOWN PLAINTEXT ATTACK MODEL

In this attack model, except the information in only ciphertext model, the cloud server knows both a set of tuples P in the attribute vectors of EMRs and the corresponding encrypted values of those tuples.

3) KNOWN BACKGROUND ATTACK MODEL

In this stronger attack model, except the information in known plaintext attack model, the cloud server is equipped with some background knowledge on the dataset. Particularly, the attacker may know some attributes values, i.e., keywords, and vector representation method. Given the background knowledge, the cloud server can speculate some attributes from the hierarchy of the index tree.

B. DESIGN GOALS

To enable secure, efficient and complex multi-attribute conjunctive keyword search over encrypted EMRs, our schemes are aiming to achieve the following main security and performance goals.

1) MULTI-ATTRIBUTE CONJUNCTIVE KEYWORD SEARCH

The scheme is expected to support complex multi-attribute conjunctive keyword search over encrypted EMRs. In practice, this type of queries is experienced in real-world applications, like patients matching, frequent itemset generating in mining association rules.

2) DYNAMIC UPDATE

The scheme aims to support dynamic update on documents collection.

3) PRIVACY GOALS

The security goals are to prevent the cloud server from learning more useful information except what can be derived from the search results. Particularly, privacy goals that we are concerned with are as follows: ① Index and query privacy: The fundamental security goal is to prevent the cloud server from learning the underlying plaintext information pertaining to the encrypted index tree and queries. ② Trapdoor unlinkability: The cloud server can't determine whether two trapdoors are generated from the same search request. ③ Keyword privacy: The cloud server can't speculate keywords and attributes from the known background knowledge on the dataset.

4) EFFICIENCY

The scheme aims to achieve better-than-linear search efficiency, by exploring a MAT based index and an efficient search algorithm over the index tree.

C. NOTATION AND PRELIMINARIES

1) NOTATION

$A-$ the attribute collection, denoted as a set of m attributes, $A = \{A_1, \ldots, A_m\}$.

TABLE 1. Original EMRs and sorted EMRs. (a) Original EMRs. (b) Sorted EMRs.

(a)

ID	sex	age	illness	region
1	male	70	hepatitis	Zhengzhou
2	female	60	diabetes	Shanghai
3	male	50	diabetes	Shanghai
4	male	70	cardiopathy	Zhengzhou
5	female	60	hectic	Zhengzhou
6	female	70	tumour	Beijing
7	female	70	tumour	Beijing
8	male	70	cardiopathy	Zhengzhou
9	male	60	tumour	Beijing
10	female	70	tumour	Shanghai

(b)

sex	age	illness	region	ID
male	50	diabetes	Shanghai	3
male	60	tumour	Beijing	9
male	70	hepatitis	Zhengzhou	1
male	70	cardiopathy	Zhengzhou	4
male	70	cardiopathy	Zhengzhou	8
female	60	diabetes	Shanghai	2
female	60	hectic	Zhengzhou	5
female	70	tumour	Beijing	6
female	70	tumour	Beijing	7
female	70	tumour	Shanghai	10

$R-$ the plaintext record collection, denoted as a set of n records, $R = \{r_1, \ldots, r_n\}$. Each record in R can be viewed as an ordered m-tuple $\{k_1, \ldots, k_m\}$ of values where k_i corresponds to attribute A_i.

$D-$the plaintext document collection, denoted as a set of n data documents, $D = \{d_1, \ldots, d_n\}$, where d_i corresponds to r_i.

$C-$the encrypted document collection stored in the cloud server, denoted as a set of encrypted data documents, $C = \{c_1, \ldots, c_n\}$, where c_i denotes the encrypted document of d_i.

$I-$ the encrypted searchable index constructed according to the record collection R.

2) PRELIMINARIES

a: MULTIPLE ATTRIBUTE TREE (MAT)

MAT is the data structure used to solve the problem of the conjunctive keyword query. An m-dimensional MAT based on m attributes of record set R is constructed as a tree with m levels. Specifically, the MAT is constructed by the following steps.

Step 1: Set the level number of MAT according to the cardinality of attribute set A, with A_i corresponding to level i. Set the level number of MAT to be the cardinality of attribute set A. Specifically, the root is at level zero and the m attributes of the records correspond to the next m levels of MAT. The root node has no a value associated with it. All other nodes of MAT are associated with the corresponding attribute values.

Step 2: Sort the records according to attribute values in $A_i (i = 1, \ldots, m)$. Sort the records in Tab. 1(a) according to attribute values, as shown in Tab. 1(b). In each column, combine all consecutive entries having the same value into a single node with this value.

Step 3: Arrange all the node keyword values in a filial set in ascending order. All the child nodes at level i derived from a single node at level (i-1) constitute a filial set at level i. All the nodes in a filial set are ranked in ascending order of their attribute values. For example, all child nodes of the node "male", i.e., "50", "60", "70" are ranked in ascending order.

Step 4: Assign records to ID array. In addition to an attribute value, any terminal node contains a physical record pointer. This pointer points to record ID array. In the record ID array, we store IDs of records, each of which has the same m attribute values.

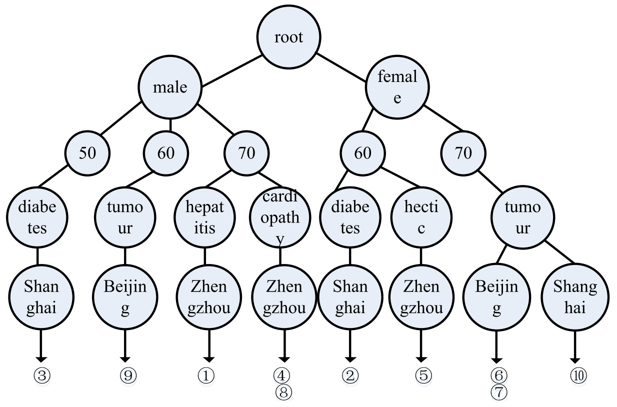

FIGURE 2. MAT representation.

According to the above steps, MAT is constructed based on the record set in Tab. 1, as depicted in Fig. 2.

b: Attribute Hierarchy and Vector Representation

As defined in the literature [11], a hierarchical attribute is constructed as a balanced tree. As a simple example of an attribute hierarchy, the numerical attribute age A_1, which has the attribute value {1,2,…,100}, is constructed as a balanced tree with 4 levels, as shown in Fig.3. Every node is assigned a unique ID. The balanced tree is seen as the set of all nodes

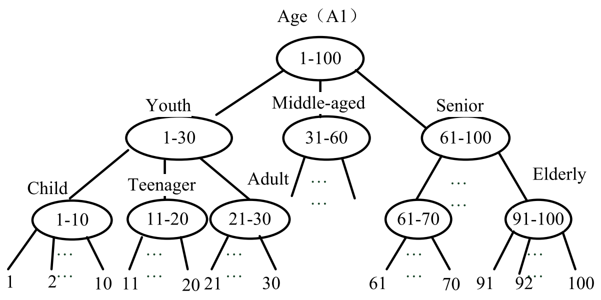

FIGURE 3. Hierarchy of numerical attribute "age".

IDs in the tree. The root node and each intermediate node represent a range, and each leaf node represents an attribute value of age, such as 50 and 60.

According to the attribute hierarchy, each attribute value can be represented as a binary vector. Any hierarchy of attribute A_i with k level can be expanded into k subfields: $A_{i,1}, \ldots, A_{i,k}$, where $A_{i,j}$ is represented by the j_{th} level construction of the tree. Specifically, $A_{i,j}$ ($j \in [1, \ldots, k]$) is represented according to the special rules.

1) $A_{i,1}$ is represented by binary 1;

2) If each node at level (j-1) ($j \in [2, \ldots, k]$) has n child nodes, then $A_{i,j}$ is represented as an n-bit binary vector. If $A_{i,j}$ is the m_{th} child node of $A_{i,j-1}$, then only the m_{th} bit is 1 and other $n-1$ bits are set to 0.

3) If the nodes at level (j-1) ($j \in [2, \ldots, k]$) have different child nodes number, then the vector length of $A_{i,j}$ is determined by the node N, which has the maximum number of child nodes. Assume that node N has p child nodes, and then $A_{i,j}$ is represented as a p-bit binary vector. If $A_{i,j}$ is the m_{th} child node of $A_{i,j-1}$, then only the m_{th} bit is 1 and other p-1 bits are set to 0.

According to the above vector representation rules, "30" is represented as 110000100000000001. Similar to the vector representation of attribute value, a query requirement can also be represented as a binary vector. For the attributes the user is not interested, the corresponding bits are all set to 1, and the other bits are set by the same method as that of attribute vectors. An attribute value range, as well as "OX" of multiple attribute values, can also be represented as binary vectors. For example, "$1 \leq$age≤ 60" and "age $= 61$ or age $= 62$" are represented as 111011111111111111 and 100110001100000000 respectively.

Based on the above vector representation, an important equation I.Q $= 0$ holds, where "." is the inner product, Q is the opposite vector of the query vector, denoted by q (e.g. if q $= 010$, then Q $= 101$), and I is the attribute vector, which satisfy the query requirement. See the paper [11] for more details.

IV. MCKS-MAT SCHEME

A. CONSTRUCTION OF UNENCRYPTED MAT INDEX

In this section, we construct the unencrypted index based on MAT construction, denoted by T. For the convenience of description, we write MAT based index as MAT index for short. If all the node keywords are encrypted, MAT index is called as encrypted MAT index. Since in the considered scenario, each record in the database has multiple attributes it is a natural idea to construct the index based on MAT. MAT-tree allows to index set of records by many attributes in one data structure. MAT-tree also supports update operation easily.

In Section III, we have briefly introduced the MAT structure, which assists us in introducing the construction of MAT index. Now we analyze the steps of constructing MAT one by one.

Step 1: Set the level number of MAT according to the number of the attributes. In this step, the level number of MAT is set to equal the cardinality of attributes set $A = \{A_1, \ldots, A_m\}$, where A_i corresponds to level i. It is improper to construct the secure MAT index directly according to the method of constructing MAT, because it is possible to deduce some attributes for the cloud server equipped with the known background knowledge. For example, if the MAT index for EMRs is constructed by step 1, the level corresponding to sex has only two nodes, thus the cloud server can speculate this level corresponds to the sex attribute. Therefore, to prevent the cloud server speculating the information from the structure of the MAT index, the original attribute set $A = \{A_1, \ldots, A_m\}$ can be strategically divided into h groups, i.e., $A' = \{A'_1, \ldots, A'_h\}$, called as the changed attribute set. Specifically, the attribute which has fewer attribute values can be strategically combined with another attribute, e.g., "sex" can be combined with other attributes, such as "age", thus the combined attribute "sex||age" has combined attribute values, such as "male ||70", "female||60" and so on. Afterwards, set the level number of MAT index to equal the number of attributes in $A' = \{A'_1, \ldots, A'_h\}$, where A'_i corresponds to level i. Therefore, step 1 is replaced by step 1'.

Step 1′: Set the level number of MAT index to equal the cardinality of the changed attribute set A'. Step 2: Sort the records according to attribute values: In this step, the records are sorted according to attribute values in the attribute set A. After A is changed into A', the records should be sorted according to attribute values in A'. Therefore, step 2 is replaced by step 2'.

Step 2′: Sort the records according to attribute values in A'_i(i=1,…,h). Step 3: Arrange all the nodes at level i descended from a single node at level (i-1) (i =1,…, m) in ascending order: In this step, all the child nodes of a node are ranked in ascending order of their attribute values. This construction is unsecured in the MAT index. If it is equipped with the background knowledge of attribute values, the cloud server can speculate node attribute values in a filial set according to the rule of ascending order. In order to avoid this speculation, we substitute step 3' for step 3.

Step 3′: Arrange all the nodes in a filial set at level i descended from a single node at level (i-1) (i=1,…,h) in random order.

When MAT index is constructed, step 4 remains unchanged.

Step 4′: Assign records to ID array.

Note, to generate the encrypted MAT index, the data owner needs to encrypt node keywords of MAT index. In MAT index, the node keywords are in the form of vectors. If the cloud server has knowledge of attribute vector representation, it is possible to speculate on attribute according to the vector length. In order to avoid this speculation, we firstly represent the node keywords at level i as a l_i-bit binary vector by the existing efficient attribute value vector representation method (In this paper, we adopt the vector representation methods proposed by Zhang *et al.* [11]), then we extend the vector to a fixed length, denoted as L, by padding 0s to the last (L-l_i) bits. In order not to add too much extra computation, L is set to be equal to or slightly more than l_x, $l_x = \max\{l_1, \ldots, l_h\}$.

B. MCKS-MAT SCHEME

1) MCKS-MAT SCHEME

Based on MAT index, we propose a multi-attribute conjunctive keyword search scheme, named as MCKS-MAT scheme. Four algorithms included are described as follows:

- ***Setup***(1^n): In the initialization phase, the data owner randomly generates L-bit vector S_i and $L \times L$ invertible matrices $\{M_{1,i}, M_{2,i}\}$, where L denotes the length of node keyword vectors. Sub-key SK_i at level i is in the form of a 3-tuple as $\{S_i,\ M_{1,i},\ M_{2,i}\}$. $SK = \{SK_1, SK_2, \ldots, SK_h\}$.
- ***GenIndex***(SK, R): Firstly, the unencrypted MAT index T is constructed on the record set R, and then the data owner encrypts the node keywords at each level. Let the vector of node keyword k_i at level i be denoted as $\vec{I_i}$. ① Multiply $\vec{I_i}$ by the random value ε_i to get $\hat{I_i}$ such that $\hat{I_i} = \varepsilon_i \cdot \vec{I_i}$, where $\varepsilon_i \neq 0$. ② Split $\hat{I_i}$ into two random vectors $\{\hat{I}_{ia}, \hat{I}_{ib}\}$. For $x = 1$ to L, if $S_i[x] = 1$, $\hat{I}_{ia}[x]$ and $\hat{I}_{ib}[x]$ are set such that $\hat{I}_{ia}[x] + \hat{I}_{ib}[x] = \hat{I_i}[x]$. If $S_i[x] = 0$, $\hat{I}_{ia}[x]$ and $\hat{I}_{ib}[x]$ are set as the same as $\hat{I_i}[x]$. ③ Generate encrypted vector $\vec{I_i}' \ .\ \vec{I_i}' = \{M_{1,i}^T \hat{I}_{ia}, M_{2,i}^T \hat{I}_{ib}\}$, denoted as $\{\vec{I}_{ia}', \vec{I}_{ib}'\}$. *Finally, the encrypted MAT index I* is constructed where the node keyword k_i exists in the form of $\{M_{1,i}^T \hat{I}_{ia}, M_{2,i}^T \hat{I}_{ib}\}$.
- ***Trapdoor***(SK, w_q): Firstly, the query q_i at level i is represented as a l_i -bit binary vector according to the query keywords set w_q. Here we adopt the query representation method introduced in Section III-C2b, which can achieve equality conjunction, range conjunction and subset conjunction. See the paper [11] for more details. Secondly, we encrypt the query vector q_i. The encryption process of q_i is as follows. ① Flip q_i bit by bit to get Q_i (e.g., if $q_i = 100$, then $Q_i = 001$). ② Extend Q_i to a L-bit vector, denoted as $\vec{Q_i}$, the last $(L - l_i)$ bits of which are padded with the random number. ③ Multiply $\vec{Q_i}$ by random value β_i to generate $\hat{Q_i} = \beta_i \cdot \vec{Q_i}$, and $\beta_i \neq 0$. ④ Split $\hat{Q_i}$ into two random vectors as $\{\hat{Q}_{ia}, \hat{Q}_{ib}\}$. For $x = 1$ to L, if $S_i[x] = 1$, $\hat{Q}_{ia}[x]$ and $\hat{Q}_{ib}[x]$ are set the same as $\hat{Q_i}[x]$; if $S_i[x] = 0$, $\hat{Q}_{ia}[x]$ and $\hat{Q}_{ib}$ are set so that $\hat{Q}_{ia}[x] + \hat{Q}_{ib}[x] = \hat{Q_i}[x]$. (5) The split data vector pair $\{\hat{Q}_{ia}, \hat{Q}_{ib}\}$ is encrypted as$\{M_{1,i}^{-1} \hat{Q}_{ia}, M_{2,i}^{-1} \hat{Q}_{ib}\}$, called as the sub-trapdoor $TD_{\vec{Q_i}}$ at level i. $TD_{\vec{Q}} = \{TD_{\vec{Q_1}}, \ldots, TD_{\vec{Q_h}}\}$.
- ***Match***$(\vec{I_i}', TD_{\vec{Q_i}})$: With this algorithm, the cloud server can test if the node keyword k_i at level i matches the query keyword corresponding to level i. The cloud server computes $\vec{I_i}' \ .\ TD_{\vec{Q_i}}$, which denotes the standard inner product. This algorithm outputs 1 if the inner product is 0, indicating this node matches the query keyword.

2) SEARCH ALGORITHM FOR MCKS-MAT SCHEME

In this section, we propose the search algorithm for the MCKS-MAT scheme, whose search process is a recursive procedure upon MAT index, named as the searchMAT algorithm. This algorithm searches the MAT index level by level from the root, collecting qualified nodes at each level. When search process enters the i_{th} level, a sequential search is executed from left to right on all the nodes in the qualified-filial-set at level i descended from a single node at level (i-1) and the qualified nodes are collected. Specifically, search process selects an unselected node from the left when it enters the i_{th} level and carries on match operation by the algorithm *Match*$(\vec{I_i}', TD_{\vec{Q_i}})$ over the encrypted node keyword k_i, here $\vec{I_i}'$ is the encryption form of the vector of keyword k_i and $TD_{\vec{Q_i}}$ is the trapdoor corresponding to level i. According to the output of *Match*$(\vec{I_i}', TD_{\vec{Q_i}})$ algorithm, search process judges whether keyword k_i matches the corresponding query keyword. That the output is 1 shows k_i matches the corresponding query keyword. At this time, search progress depends on whether k_i resides at the bottom level. If k_i resides at the bottom level, at which the pointers indicate record ID array, search process appends record IDs from record id array to the result list T. If k_i resides at the intermediate level, search process goes down to the next level and carries out sequential search from left to right on all nodes in the filial-set. When the output of the algorithm *Match*$(\vec{I_i}', TD_{\vec{Q_i}})$ is not 1, The search process selects the right node of the node k_i. The algorithm *searchMAT* is shown in Fig. 4. Note, in this paper, we adopt the vector representation method proposed by Zhang et al. [11], Therefore the above match has the following meanings: ① the user doesn't care attribute at this level; ② this node keyword equals to the query keyword corresponding to this level; ③ this node keyword belongs to the query range corresponding to this level; ④ this node keyword is the subset of the query corresponding to this level.

To describe *searchMAT* algorithm explicitly, we use the order of node keywords to denote the identifier of the filial-sets at level i descended from a single node at level (i-1). The identifier of the root node is denoted as (∅) and the identifier of the filial-sets at level i, descended from a single node at level (i-1), is denoted as $(k_1, \ldots, k_{i-1})$, e.g., in Fig. 2, $(k_1, k_2) = $ (male, 70) is the identifier of the filial set in the third level, which contains the node keyword "hepatitis" and "cardiopathy".

```
searchMAT algorithm
Input: MAT MAT index, Trapdoor $TD_{\overline{Q}}$;
Output: List L;
var List L;  { list of record IDs}
begin
searchMAT (MAT index, ( ¢ ), $TD_{\overline{Q}}$);
return L;
end.
procedure searchMAT (MAT MAT index, Identifier (k1, ..., ki-1), Trapdoor $TD_{\overline{Q_i}}$ );
while(there is another node in the filial-set) do
In the filial set of MAT index with the identifier (k1, ..., ki-1)
ki=get NextKeyword(MAT index, (k1, ..., ki-1);
Pd= $Test(\overrightarrow{I_i}', TD_{\overline{Q_i}})$ // here $\overrightarrow{I_i}'$ is the encryption form of vector of keyword ki
if ((Pd =1) then  // ki matches with the query keyword
            If keyword ki resides at intermediate level then
               searchMAT (MAT-index, (k1, ..., ki), $TD_{\overline{Q_{i+1}}}$ );
         If ki resides at the bottom level then
            Append record IDs from this array to the list L;
if (Pd ≠1) then // ki doesn't match with the query keyword
         if(ki is the last keyword in filial set) then
            searchMAT (MAT-index, (k1, ..., ki-2), $TD_{\overline{Q_{i-1}}}$ ));//return the filial set in upper level
         else
         get NextKeyword(MAT index, (k1, ..., ki-1)
endwhile
end
Procedure getNextKeyword (MAT MAT index, Identifier (k1, ..., ki-1))
In filial set of MAT index with identifier(k1, ..., ki-1) choose the next keyword ki toward the right
Return ki;
end
```

FIGURE 4. searchMAT algorithm.

3) CORRECTNESS ANALYSIS OF MCKS-MAT SCHEME

$$\overrightarrow{I_i}' \cdot TD_{\overrightarrow{Q_i}} = \{M_{1,i}^T \hat{I}_{ia}, M_{2,i}^T \hat{I}_{ib}\}. \{M_{1,i}^{-1} \hat{Q}_{ia}, M_{2,i}^{-1} \hat{Q}_{ib}\} = M_{1,i}^T \hat{I}_{ia}.\ M_{1,i}^{-1} \hat{Q}_{ia} + M_{2,i}^T \hat{I}_{ib}.\ M_{2,i}^{-1} \hat{Q}_{ib} = \varepsilon_i.\ \beta_i \sum_{n=1}^{L} \overrightarrow{I_i}[n].\overrightarrow{Q_i}[n] = \varepsilon_i.\ \beta_i(\sum_{n=1}^{l_i} \overrightarrow{I_i}[n].\overrightarrow{Q_i}[n] + \sum_{n=l_i+1}^{L} \overrightarrow{I_i}[n].\overrightarrow{Q_i}[n])$$

It is obvious that $\sum_{n=l_i+1}^{L} \overrightarrow{I_i}[n].\overrightarrow{Q_i}[n]$ is 0, because the (L-l_i) bits of $\overrightarrow{I_i}$ are all set to 0. When node keyword k_i at level i matches the query keyword at this level, the formula $\sum_{n=1}^{l_i} \overrightarrow{I_i}[n].\ \overrightarrow{Q_i}[n]$ is 0 based on the vector representation method in Section III-C2b. As a result, when node keyword k_i at level i matches the query keyword at this level, $\varepsilon_i.\beta_i \sum_{n=1}^{L} \overrightarrow{I_i}[n].\overrightarrow{Q_i}[n]$ is 0, i.e., the inner product $\overrightarrow{I_i}' \cdot TD_{\overrightarrow{Q_i}} = 0$, otherwise, $\overrightarrow{I_i}' \cdot TD_{\overrightarrow{Q_i}} \neq 0$.

4) SECURITY ANALYSIS OF MCKS-MAT SCHEME

MCKS-MAT CAN RESIST AGAINST THE KNOWN PLAINTEXT ATTACK:

Proof: Suppose that an attacker knows both a set of tuples P in the database and the corresponding encrypted values of those tuples. Without loss of generality, we assume no random number is introduced (Note that introduction of random number will enhance the security of the scheme). For any vector $\overrightarrow{I_i} \in P$, by the scheme, a known plaintext attacker knows the encrypted values $\{\overrightarrow{I_{ia}}', \overrightarrow{I_{ib}}'\}$). If the attacker does not know the splitting configuration, he has to model $\overrightarrow{I_{ia}}$ and $\overrightarrow{I_{ib}}$ as two random n-dimensional vectors. The equations for solving the transformation matrices are $\overrightarrow{I_{ia}}' = M_{1,i}^T \overrightarrow{I_{ia}}$ and $\overrightarrow{I_{ib}}' = M_{2,i}^T \overrightarrow{I_{ib}}$, where $M_{1,i}$ and $M_{2,i}$ are two $L \times L$ unknown invertible matrices. The number of equations is $2L\,|P|$, where $|P|$ is the cardinality of the set P, however, there are $2L\,|P|$ unknowns in $\overrightarrow{I_{ia}}$ and $\overrightarrow{I_{ib}}$, and $2L^2$ unknowns in $M_{1,i}$ and $M_{2,i}$. Obviously, the attacker has no sufficient information to solve for $M_{1,i}$ and $M_{2,i}$. Hence, MCKS-MAT scheme can withstand the known plaintext attack.

In the following, we analyze the scheme according to the three predefined security requirements. ① Index and query privacy: Index privacy mainly refers to the privacy of node keyword vectors. As long as the secret key is not revealed, the underlying plaintext information in the keyword vector and query vector is well protected, as shown in [31]. Moreover, nonlinear vector splitting module is adopted and random numbers ε_i and β_i are introduced into attribute vector and query vector. ② Trapdoor unlinkability: as a result of random vector splitting process and the introduction of the random number β, the adopted query vector encryption method provides non-deterministic encryption. The trapdoor generation algorithm outputs two different trapdoors for the same query request, which guarantees trapdoor unlinkability. ③ Keyword privacy: in only ciphertext model and known plaintext model, the keyword is well protected, however, in known background model, the cloud server may know some backgrounds knowledge about the database, such as some attribute values, i.e., keywords. To prevent the cloud server speculating keywords from the structure of the MAT index, we strategically divide the original attribute set such that the attribute with fewer values is combined with other attributes. In addition to this, we arrange all the node keyword in a filial set at level i descended from a single node at level (i-1) in random order. Moreover, to avoid deducing attributes from the length of keyword vectors, we represent all the node keywords and query keywords with a fixed vector length. Therefore the cloud server has no way to distinguish the attribute at each level.

Like most of existing encrypted search schemes, our search scheme sacrifices access pattern privacy for efficiency.

V. DYNAMIC UPDATE ALGORITHM

Similar to the schemes in [14]–[16], we only consider insertion and deletion of documents. After insertion or deletion of a document, the index needs to be updated synchronously. The index of the proposed schemes is constructed as a MAT construction; therefore the update operation is performed by updating the corresponding nodes in MAT index. Note that the update on the index is merely based on document identifies (ID), and it isn't needed to access the content of documents. The update on the index is described as follows:

A. GenUpdataInfo(SK, T_s, i, updtype)

Here i refers to the identifier of the record r_i. The notion $updtype \in \{del, ins\}$ denotes either a deletion or an insertion for the record r_i. The notion T_s denotes the subtree consisting of the tree nodes and ID array that need to be changed during the update operation. For example, if we want to delete the record r_1 in Fig. 2, T_s refers to the set {hepatitis, Zhengzhou, ID array:{ID = 1}}.

Deletion is discussed in two cases. If one or more records in the record set have the same m attribute value as the deleted record r_i, simply delete r_i's ID from the original ID array. In the other case, it needs to analyze each level keyword of the deleted record r_i. For the convenience of description, we denote the nodes by the node keywords. For example, we want to delete the record r_1 in Fig. 2. In the record set there exist many records whose first level keyword is male, therefore the node "male" remains unchanged. In the record set there exist many records with the keywords {male, 70}, therefore the node "70" remains also unchanged. However, the record r_1 is the only one that satisfies that the top three level keywords are {male, 60, hepatitis}; hence the node "hepatitis" needs to be deleted. Similarly, it needs to delete the node "Zhengzhou" and the ID array where ID = 1 is stored. Finally, c_i is set to null. Here c_i is the encrypted document of d_i corresponding to the record r_i.

Insertion is also discussed in two cases. If one or more records in the record set have the same m keywords as the inserted record r_i, simply insert r_i's ID into the original ID array. In the other case, it need analyze each level keyword of inserted record r_i. Take Fig. 2 as an example, we want to insert the record r_{11} with keywords {female, 60, tumour, Zhengzhou}. In Fig. 2 there exist the record r_2 and r_5 whose front two level keywords are {female, 60}, therefore the nodes "female" and "60' remain unchanged. In Fig. 2 exists no record whose front three level keywords are {female, 60, tumour}, so it needs to insert the node "tumour" as the child node of "60". Similarly, it is necessary to insert "Zhengzhou" as the child node of "tumour". The pointer of "Zhengzhou" points to a newly inserted ID array containing only ID = 11. The data owner encrypts the vectors of the newly inserted node keywords with the secret key SK to generate encrypted subtree $T_s^{'}$. The encryption of node keyword vectors is introduced in Section IV-B. Finally, the record d_i is encrypted to c_i by the encryption method of documents content. Note, as a result of introducing the random number in the encryption process, encryption is an indeterminate encryption such that the same keyword is encrypted into different ciphertext.

B. Updata(I, C, updtype, $T_s^{'}$, c_i)

After receiving the update information from the data owner, the cloud server carries out the update operation. When $updtype$ is del, the cloud server directly deletes $T_s^{'}$ (the encrypted form of T_s) from the index structure stored in the cloud so as to generate a new index tree I' and delete c_i from the encrypted document collection C. If $updtype$ is ins, the cloud server inserts the encrypted document c_i into C, obtaining a new ciphertext collection C', in addition, it inserts a new subtree T_S' to generate a new index structure I'. To enable the cloud server to know where the new subtree is inserted or deleted, it is necessary to assign an identifier for every node when the MAT index is constructed.

Similar to the scheme in [16], we store an unencrypted tree on the data owner side. Then, the data owner can update the subtree directly according to the records newly inserted or deleted, encrypt and upload the updated subtree to the cloud server.

Generally, patients have fixed attribute values for a large database, therefore it is reasonable to represent vectors with a fixed length.

VI. COMPARISON OF PERFORMANCE

To evaluate the whole performance of MCKS-MAT scheme, we compare MCKS-MAT scheme with MCKS_II scheme proposed by Zhang *et al.* [11], which is a linear search scheme. Let n be the number of records in R, m the number of attributes in A.

A. PREPROCESSING COST

We use P (n, m) to denote the cost of preprocessing n records into a data structure. The preprocessing cost for MCKS_II scheme is $P_{MCKS\_II}(n, m) = O(mn)$, as a result of constructing the tree-based index, the preprocessing cost for MAT in MCKS-MAT scheme is higher, i.e, $P_{MAT}(n, m) = \mathrm{O}(\mathrm{mn} \log \mathrm{n} + \mathrm{mn})$,as given in [32].

B. STORAGE COST

We use $S\ (n,\ m)$ to denote the storage required to store n records. Storage for MCKS_II scheme is $S_{MCKS\_II}\ (n,\ m) = O(mn)$ *and* Storage for MAT in MCKS-MAT $S_{MAT}(n,m) = O(mn_c) = O(mn),$ where c is a constant (normally $c = 3$), as given in [32]

In the following, we compare the computation overheads of MCKS-MAT scheme and MCKS_II scheme [11]. We implement the entire secure search system using java on win7 server with Inter i5-5200U@3.30Ghz, 3.30Ghz. Since there are yet no EMRs databases publicly available for research purposes, we carry out a proof-of-concept performance demonstration of our solutions. The dataset used features categorical attributes and has 4 attributes {age, sex, illness, region}. Sex has two attribute values {female, male}. Assume that "age", "illness" and "region" have 100 attribute values respectively and the vector representation method proposed in [11] is adopted here. Based on this vector representation, sex is represented as 2-bit vector and age, illness and region are all represented as 18-bit vectors.

C. INITIALIZATION

In the initialization of MCKS_II scheme, the main overhead is the time to generate an l-bit binary vector S, as well as two $l \times l$ invertible matrices M_1 and M_2, where l is the dimensionality of the index or query vectors. Beyond this, the overhead in the initialization phase includes the time spent to pre-compute the transpose of the matrices and the inverse of the matrices, i.e., M_1^T, M_2^T, M_1^{-1} and M_2^{-1}. In MCKS-MAT scheme, on account of the tree-based index, the data owner needs to generate sub-key corresponding to each level. Specifically, the data owner needs to generate L-bit vector S_i and $L \times L$ invertible matrices $\{M_{1.i}, M_{2,i}\}$, as well as $M_{1,i}^T$, $M_{2,i}^T$, $M_{1,i}^{-1}$ and $M_{1,i}^{-1}$, where L is the fixed vector length. In MCKS-MAT scheme, assume that we combine sex with age and the combined attribute resides at the first level, and that illness and region reside at the second and third level respectively, then the length of the vector at the first level is 20-bit, and the length of the vector at the second and third levels is 18-bit. We set the fixed vector length to 20-bit, thus S_i becomes 20-bit long, and $M_{1.i}$ and $M_{2,i}$ become 20×20 invertible matrices. Testing 1000 times, the average setup time of MCKS_II and MCKS-MAT scheme is 0.634ms and 0.217ms respectively. Obviously, MCKS-MAT scheme consumes less setup time.

D. CONSTRUCTION OF ENCRYPTED INDEX

In MCKS_II scheme, the process of constructing an encrypted index falls into two steps: the first step is to map the keyword set extracted from each record to a binary vector and

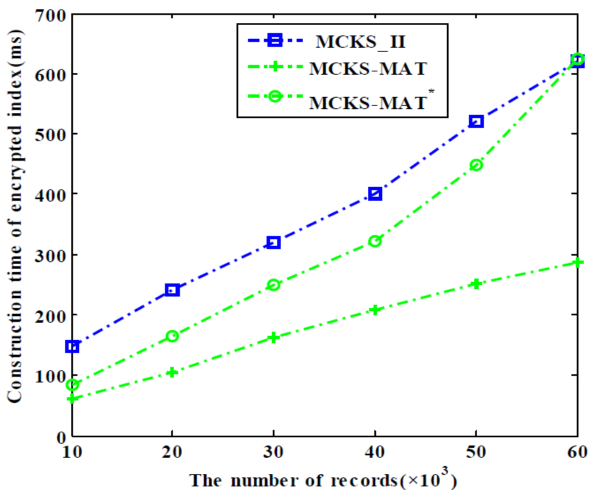

FIGURE 5. The time of constructing the encrypted index. Curve MCKS-MAT indicates that construction time of index tree isn't counted and only the time of encrypting node keywords is counted, but MCKS-MAT* indicates that construction time of index tree is counted.

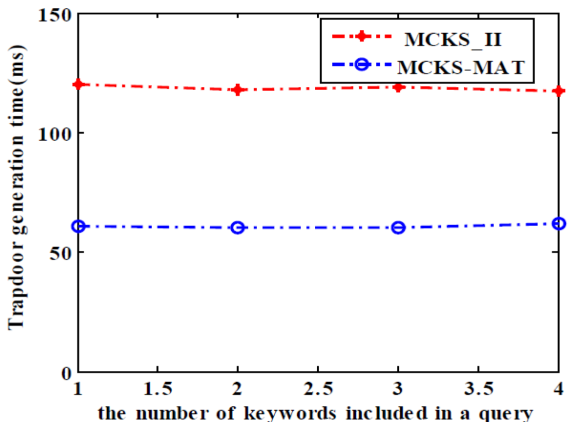

FIGURE 6. The time of generating trapdoors.

the second one is to encrypt every binary vector. The time cost is principally affected by the cardinality of record collection and dimensionality of index vectors. The latter is determined by both the distinct keywords in the record collection and attributes structure (whether the hierarchy is adopted and what the hierarchical structure is). However, in addition to the two steps performed in MCKS_II scheme, the encrypted MAT index needs to be constructed in MCKS-MAT. In the experiments, we omit the mapping time in the first step. Construction of encrypted index is discussed in two cases. When the time of generating MAT index isn't contained and only the time of encrypting node keywords is counted, MCKS-MAT scheme assumes less time. Otherwise, with the increasing of the size of the record collection, it is possible for MCKS-MAT scheme to consume more time than MCKS_II scheme, as shown in Fig. 5. Slight time increasing in generating encrypted index is acceptable, because this operation is a one-time operation.

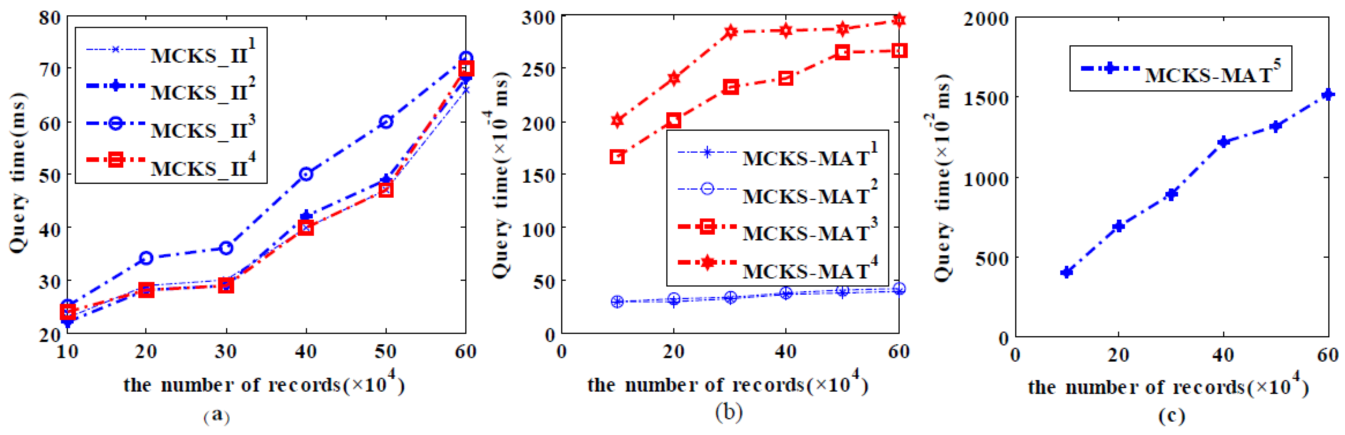

FIGURE 7. **Query time. (a) the curve MCKS_II^1 (MCKS_II^2, MCKS_II^3, MCKS_II^4) indicates the front two (the front three, all four, the last three) attributes are concerned in the query of MCKS_II scheme; (b) the curve MCKS_MAT^1 (MCKS_MAT^2, MCKS_MAT^3, MCKS_MAT^4) indicates the front two (the front three, all four, the last three) attributes are concerned in the query of MCKS_MAT scheme; (c)the curve MCKS_MAT^5 indicates the last two keywords are concerned in the query of MCKS_MAT scheme.**

E. TRAPDOOR GENERATION

In MCKS_II scheme, trapdoor generation contains a vector splitting and two multiplications of a $l \times l$ matrix, where l is the dimensionality of index or query vectors. In MCKS-MAT scheme, generation of every sub-trapdoor corresponding to every level is similar to the operation in MCKS_II scheme. Fig. 6 shows, the number of keywords contained in a query has little influence on the overhead of trapdoor generation, because the length of query vectors is constant in both MCKS_II and MCKS-MAT schemes. For the sake of explicit drawing, we test the time to generate 10000 trapdoors, as shown in Fig. 6, which illustrates that MCKS-MAT scheme consumes almost half of the time of trapdoor generation in MCKS_II scheme.

F. QUERY TIME

Fig. 7 (a) shows that the query time of MCKS_II scheme is hardly affected by the number of query keywords and is determined by the size of record collection; however, from Fig. 7(b) and Fig. 7(c) the query time in the proposed scheme is dominated by not only the size of record collection but also the number of query keywords and the position the query keywords reside. The more the records are, the more the query time consumed becomes; the lower level the query keywords reside in, the more the query time consumed is. Fig. 7 shows that MCKS-MAT achieves much higher efficiency than linear search in MCKS_II scheme. In the best case of this experiment, i.e., all four attributes are concerned, the search efficiency of the scheme proposed is increased by 2 to 3 orders of magnitude, and in the worst case, i.e., the only last two attributes are concerned, the efficiency is increased by about one orders of magnitude.

VII. CONCLUSION

In this paper, we present a secure multi-attribute conjunctive keyword search scheme over encrypted cloud data, which simultaneously supports insertion and deletion operation of documents. We construct a special MAT based index and propose an efficient search algorithm over MAT index, names as the *searchMAT* algorithm. As a result of the adoption of the special MAT-based index structure and the *searchMAT* algorithm, the proposed scheme can achieve better-than-linear search efficiency and deal with the deletion and insertion of documents flexibly. Extensive analysis and experiment results demonstrate that the proposed schemes are extremely practical.

REFERENCES

[1] M. Ali, S. U. Khan, and A. V. Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci.*, vol. 305, pp. 357–383, Jun. 2015.
[2] C. Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Palo Alto, CA, USA, 2009.
[3] Y. Wang, J. Wang, and X. Chen, "Secure searchable encryption: A survey," *J. Commun. Inf. Netw.*, vol. 1, no. 4, pp. 52–65, Dec. 2016.
[4] Z. R. Shen, W. Xue, and J. W. Shu, "Survey on the research and development of searchable encryption schemes," *J. Softw.*, vol. 25, no. 4, pp. 880–895, Jan. 2014.
[5] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. Int. Conf. Appl. Cryptog. Netw. Secur. (ACNS)*, Heidelberg, Germany, 2004, pp. 31–45.
[6] R. Brinkman, P. Hartel, and W. Jonker, "Conjunctive wildcard search over encrypted data," in *Proc. Workshop Secure Data Manag. (SDM)*, Seattle, WA, USA, vol. 2011, pp. 114–127.
[7] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Theory of Cryptography*. San Francisco, CA, USA: Springer, 2009, pp. 457–473.
[8] M. Li, S. Yu, N. Cao, and W. Lou, "Authorized private keyword search over encrypted data in cloud computing," in *Proc. Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Minneapolis, MO, USA, Jun. 2011, pp. 383–392.
[9] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptograph. Conf. (TCC)*, Amsterdam, The Netherlands, 2007, pp. 535–554.
[10] 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. EUROCRYPT*, Riviera, France, 2010, pp. 62–91.
[11] L. L. Zhang, Y. Q. Zhang, X. F. Liu, and H. Y. Quan, "Efficient conjunctive keyword search over encrypted medical records," (in Chinese), *J. Softw.*, vol. 27, no. 6, pp. 1577–1591, Jun. 2016.

[12] W. Sun *et al.*, "Privacy-Preserving multi-keyword text search in the cloud supporting similarity-based ranking," in *Proc. ASIACCS*, Hangzhou, China, 2013, pp. 71–82.

[13] W. Sun, W. Lou, Y. T. Hou, and H. Li, "Privacy-Preserving keyword search over encrypted data in cloud computing," in *Secure Cloud Computing*. New York, NY, USA: Springer-Verlag, 2014, pp. 189–212.

[14] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Syst. (CCS)*, Raleigh, NC, USA, 2012, pp. 965–976.

[15] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Proc. Int. Conf. Financial Cryptogr. Data Secur. (FC)*, 2013, pp. 258–274.

[16] Z. Xia, 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, Jan. 2016.

[17] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy (SP)*, Berkeley, CA, USA, May 2000, pp. 44–55.

[18] 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, Jan. 2011.

[19] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, "Zerber+R: Top-k retrieval from a confidential index," in *Proc. Int. Conf. Extending Database Technol., Adv. Database Technol. (EDBT)*, New York, NY, USA, 2009, pp. 439–449.

[20] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proc. Int. Conf. Distrib. Comp. Syst. (ICDCS)*, Genoa, Italy, Jun. 2010, pp. 253–262.

[21] 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.

[22] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*, Interlaken, Switzerland, 2004, pp. 506–522.

[23] L. Ballard, S. Kamara, and F. Monrose, "Achieving efficient conjunctive keyword searches over encrypted data," in *Proc. Int Conf. Inf. Commun. Secur. (ICICS)*, Beijing, China, 2005, pp. 414–426.

[24] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Proc. Int. Workshop. Inf. Secur. Appl. (WISA)*, Jeju Island, South Korea, 2004, pp. 73–86.

[25] Y. H. 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.*, Tokyo, Japan, 2007, pp. 2–22.

[26] C. Liu, L. Zhu, and J. Chen, "Efficient searchable symmetric encryption for storing multiple source data on cloud," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, Aug. 2015, pp. 451–458.

[27] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn. (EUROCRYPT)*, Istanbul, Turkey, 2008, pp. 146–162.

[28] T. Okamoto and K. Takashima, "Hierarchical predicate encryption for inner-products," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, Tokyo, Japan, 2009, pp. 214–231.

[29] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 829–837.

[30] W. Zhang, S. Xiao, Y. Lin, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, May 2016.

[31] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. Int. Conf. Manage. Data*, Providence, RI, USA, 2009, pp. 139–152.

[32] S. V. N. Rao, S. S. Iyengar, and C. E. V. Madhavan, "A comparative study of multiple attribute tree and inverted file structures for large bibliographic files," *Inf. Process. Manage.*, vol. 21, no. 5, pp. 433–442, Jan. 1985.

LILI ZHANG received the M.S. degree in cryptography from Xidian University in 2008. She currently is pursuing the Ph.D. degree with the National Key Laboratory of Integrated Services Networks, Xidian University. Her current research interests include security protocol and cloud computing.

YUQING ZHANG received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2000. He is currently a Professor in computer sciences and the Director of the National Computer Network Intrusion Protection Center, University of Chinese Academy of Sciences. His research interests include network and system security, cryptography, and networking.

HUA MA received the B.S. and M.S. degrees in mathematics from Xidian University in 1985 and 1990, respectively. She is currently a Professor with Xidian University. Her research interests include applied cryptography and cloud computing security.