Verifiable Privacy-Preserving Multi-Keyword Text Search in the Cloud Supporting Similarity-Based Ranking

Wenhai Sun, *Student Member, IEEE*, Bing Wang, *Student Member, IEEE*, Ning Cao, *Member, IEEE*, Ming Li, *Member, IEEE*, Wenjing Lou, *Senior Member, IEEE*, Y. Thomas Hou, *Fellow, IEEE*, and Hui Li, *Member, IEEE*

Abstract—With the growing popularity of cloud computing, huge amount of documents are outsourced to the cloud for reduced management cost and ease of access. Although encryption helps protecting user data confidentiality, it leaves the well-functioning yet practically-efficient secure search functions over encrypted data a challenging problem. In this paper, we present a verifiable privacy-preserving multi-keyword text search (MTS) scheme with similarity-based ranking to address this problem. To support multi-keyword search and search result ranking, we propose to build the search index based on *term frequency* and the *vector space model* with *cosine similarity measure* to achieve higher search result accuracy. To improve the search efficiency, we propose a tree-based index structure and various adaptive methods for *multi-dimensional (MD) algorithm* so that the practical search efficiency is much better than that of linear search. To further enhance the search privacy, we propose two secure index schemes to meet the stringent privacy requirements under strong threat models, i.e., known ciphertext model and known background model. In addition, we devise a scheme upon the proposed index tree structure to enable authenticity check over the returned search results. Finally, we demonstrate the effectiveness and efficiency of the proposed schemes through extensive experimental evaluation.

Index Terms—Cloud computing, privacy-preserving search, multi-keyword search, similarity-based ranking, verifiable search

1 Introduction

Cloud computing is a new model of enterprise IT infrastructure that enables ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) [2]. Due to the centralized management of elastic resources, all players in this emerging X-as-a-service (XaaS) model, including the cloud provider, application developers, and end-users, can reap benefits. Especially, for the end-users, they can outsource large volumes of data and workloads to the cloud and enjoy the virtually unlimited computing resources in a pay-per-use manner. Indeed, many companies, organizations, and individual users have adopted the cloud platform to facilitate their business operations, research, or everyday needs [3].

Despite the tremendous business and technical advantages, privacy concern is one of the primary hurdles that prevent the widespread adoption of the cloud by potential users, especially if their sensitive data are to be outsourced to and computed in the cloud. Examples may include financial and medical records, and social network profiles. Cloud service providers (CSPs) usually enforce users' data security through mechanisms like firewalls and virtualization. However, these mechanisms do not protect users' privacy from the CSP itself since the CSP possesses full control of the system hardware and lower levels of software stack. There may exist disgruntled, profiteered, or curious employees that can access users' sensitive information for unauthorized purposes [4], [5]. Although encryption before data outsourcing [6], [7] can preserve data privacy against the CSP, it also makes the effective data utilization, such as search over encrypted data, a very challenging task. Without being able to extract useful information from the outsourced data in a secure and private manner, the cloud will merely be a remote storage which provides limited value to all parties.

One fundamental and common form of data utilization is the search operation, i.e., to quickly sort out information of interest from huge amount of data. The information retrieval community has the state-of-the-art techniques that are readily available to achieve rich search functionalities, such as result ranking and multi-keyword queries, on plaintext. For example, *cosine measure* in the *vector space model* [8] is a state-of-the-art similarity measure widely used in plaintext information retrieval, which incorporates the ''term frequency (TF) $\times$ inverse document frequency

- *W. Sun is with the State Key Laboratory of Integrated Services Networks, Xidian University, China and Virginia Polytechnic Institute and State University, USA. E-mail: whsun@xidian.edu.cn.*
- *B. Wang, W. Lou, and Y.T. Hou are with Virginia Polytechnic Institute and State University, USA. E-mail: {bingwang, wjlou, thou}@vt.edu.*
- *N. Cao was with Worcester Polytechnic Institute, USA and now with Google Inc. E-mail: ncao@wpi.edu.*
- *M. Li is with Utah State University, USA. E-mail: ming.li@usu.edu.*
- *H. Li is with the State Key Laboratory of Integrated Services Networks, Xidian University, China. E-mail: lihui@mail.xidian.edu.cn.*

Manuscript received 11 July 2013; revised 20 Oct. 2013; accepted 22 Oct. 2013. Date of publication 5 Nov. 2013; date of current version 15 Oct. 2014. Recommended for acceptance by H. Shen.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.282

(IDF)'' weight to evaluate the similarity between a document and a particular query, and yield accurate ranked search result. However, implementing a secure version of such techniques over outsourced encrypted data in the cloud is not straightforward, and is susceptible to privacy breach [9]. Although inverted index (a.k.a. inverted file) is the most popular and efficient index data structure used in document retrieval systems, it is not directly applicable in TF-based multi-keyword encrypted text search environment [9], [10], [11], since similarity scores cannot be order preserving when query involves multiple keywords (we chose to only give a very brief explanation that provides root cause but not the details).

1.1 Related Work

In the literature, searchable encryption (SE) techniques [12], [13], [14], [15], [16], [17], [18], [19], [20], [21] can partially address the need for secure outsourced data search as follows.

1.1.1 Searchable Encryption With Single Keyword

Song *et al.* [16] proposed the first SE scheme, where, to search a certain keyword in a document, user has to go through the whole document. After this work, many improvements and novel schemes [17], [18], [19], [20] have been proposed. Curtmola *et al.* [19] proposed an inverted index based SE scheme with extremely efficient search process, but the *keyword privacy* will be revealed if the corresponding keywords have been searched. Frequency information was not involved in the similarity evaluation processes of the above techniques to provide accurate search functionality. In [9], [10], [11], the order-preserving techniques were utilized to protect the sensitive frequency related information. Due to the index and query built from frequency related information and the inverted index as the underlying index structure, they can achieve accurate and efficient search at the same time. Boneh *et al.* [12] proposed the first PKC-based SE scheme, where anyone with public key can write to the data stored on server but only authorized users with private key can search. However, all of the aforementioned solutions only support single keyword search.

1.1.2 Searchable Encryption With Multiple Keywords

In the public key setting, a lot of works have been done to realize the conjunctive keyword search, subset search, or range queries [13], [14], [15], but they are too computationally intensive to be implemented for practical use. Predicate encryption is another promising technique to fulfill the search over encrypted data [22]. In [23], a logarithmic-time search scheme was presented to support range queries, which is orthogonal to our text search scenario. In text retrieval scenario, Pang *et al.* [24] proposed a vector space model based secure search scheme. An access manager is supposed to exist in their protocol except for a document server, and additional overhead occur on the user side. Without the security analysis for frequency information in their scheme, it is not clear whether such sensitive information disclosure could lead to *keyword privacy* infringement. Besides, the practical search performance is absent from the demonstration of their experiment. Cao *et al.* [21] proposed a privacy-preserving multi-keyword ranked search scheme. Although with ''coordinate matching'', this scheme can produce the ranked search result by the number of matched keywords, more accurate ranked search result is not considered there, and the search complexity is constant in that the cloud server has to traverse all the indexes of the document set for each search request.

None of the aforementioned works can achieve accurate and efficient multi-keyword secure text search supporting similarity-based ranking simultaneously.

1.1.3 Verifiable Search Based on Authenticated Index Structure

Due to possible software/hardware failure, storage corruption, etc., cloud server may return erroneous or false search results. Search result verification is a desirable feature that a robust search system would like to provide to its users.

In the plaintext database scenario, verifiable search functionality has been studied extensively since the outsourced database model emerged, e.g., [25], [26]. Most of these works adopted Merkle hash tree and cryptographic signature techniques to construct authenticated tree structure upon which end users can verify the correctness and completeness of the query results. One the other hand, Pang *et al.* [27] also used Merkle hash tree based authenticated structure to enable verification function over the query results generated by text search engines. Similar to the works in the database, they only considered the verification-specific issues regardless of the search privacy preserving capabilities that we provide in this paper.

In encrypted data search scenario, Wang *et al.* [9] used hash chain to construct a single keyword search result verification scheme. In [23], the author adapted the verification techniques in plaintext database to the encrypted database but it is not applicable to our encrypted text search scenario.

Thus, the quest for secure data search mechanisms that can simultaneously achieve high efficiency and functionality (such as expressive/usable queries) still remains open up to date.

1.2 Our Contributions

In this paper, we address the challenges of constructing practically efficient and flexible encrypted data search functionalities that support multi-keyword queries, result ranking and result verification. In particular, to support multi-keyword queries and search result ranking functionalities, we propose to build the search index based on the vector space model and adopt the cosine similarity measure that incorporates the $TF \times IDF$ weight for higher search accuracy. To improve the search efficiency, we divide the long vector index into multiple layers and propose a tree-based index structure, where each value in a node is a sub-vector from the long index vector. We then apply the search algorithm, adapted from the MD-algorithm [28], so as to realize more efficient search functionality. Our basic scheme for multi-keyword text search with similarity-based ranking (BMTS) is secure under the known ciphertext model. To further enhance the search privacy, we propose another enhanced secure index scheme (EMTS) against sensitive frequency information

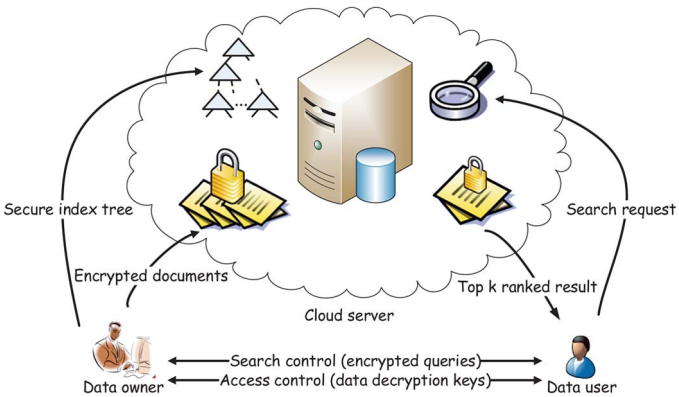


Fig. 1. Framework of the search over outsourced encrypted cloud data.

leakage to meet more stringent privacy requirements under a stronger threat model, i.e., known background model. Furthermore, we devise a scheme to enable the authenticity check over the returned search results by using hash and signature techniques. Finally, we demonstrate the effectiveness and efficiency of the proposed schemes through extensive experimental evaluation. Our contributions are summarized as follows:

1. By incorporating the state-of-the-art information retrieval techniques, we propose a privacy-preserving multi-keyword text search scheme supporting similarity-based ranking, which enjoys the same flexibility and search result accuracy as the state-of-the-art multi-keyword search over plaintext.
2. We propose a randomization (phantom terms) approach in the enhanced scheme to prevent sensitive frequency information leakage thus achieving better privacy of keywords. We show that with the proposed methods, user can balance between search precision and privacy.
3. With improved security guarantee, EMTS is still comparable in search time to BMTS. In addition, we investigated various index building methods to speed up the search of common cases. The results demonstrate much improved search efficiency compared with [21].
4. Upon the proposed index tree structure, we present a mechanism to help users ensure the authenticity of the returned search results in the multi-keyword ranked encrypted text search scenario.

The remainder of this paper is organized as follows. In Section 2, we present the system formulation. Then we elaborate on the secure index schemes in Section 3. We discuss various index building methods to accelerate the generic search process in Section 4 and deal with the search result verification in Section 5. Section 6 presents the intensive performance evaluation. Finally, we draw the conclusion in Section 7.

2 Problem Formulation

2.1 System Model

The system model considered in this paper consists of three entities: the *data owner*, the *data user*, and the *cloud server*, as illustrated in Fig. 1. The data owner outsources a huge size of document collection $\mathcal{DC} = \{d|d_1, d_2, \ldots, d_m\}$ in the encrypted form $\mathcal{C} = \{c|c_1, c_2, \ldots, c_m\}$, together with an h-level searchable index tree $\mathcal{I}$ generated from $\mathcal{DC}$, to the cloud server. We assume that the data user has the mutual authentication capability with the data owner. As such, search control mechanisms can be applied here, e.g., broadcast encryption [19], through which the data user obtains the encrypted search query $\widetilde{Q}$. Upon the receipt of $\widetilde{Q}$, the cloud server starts searching the index tree $\mathcal{I}$ and will return the corresponding set of encrypted documents, which have been well-ranked by our frequency based similarity measures (as will be introduced shortly). The data user may also send a search parameter k along with the search query $\widetilde{Q}$ such that the cloud server only returns the top-k most relevant documents. The capability of the user to decrypt the received documents [6], [7] is a separate issue and is out of the scope of this paper.

2.2 Threat Model

We assume that the cloud server acts in an ''honest-but-curious'' manner, which is also employed by related works on secure cloud data search [9], [21]. In other words, the cloud server honestly follows the protocol execution, but curiosity propels him/her to the speculation and analysis over the data and searchable index tree available at the server. Depending on the available information to the cloud server, two threat models are considered here.

2.2.1 Known Ciphertext Model

Only the encrypted document set $\mathcal{C}$, searchable index tree $\mathcal{I}$ and encrypted query vector $\widetilde{Q}$, all of which are outsourced from the data owner, are available to the cloud server. Specifically, we intend to protect the plaintext query/index information against the cloud server and keep the dictionary of n keywords $\mathcal{T} = \{t|t_1, t_2, \ldots, t_n\}$ as secret that was used to build the index tree $\mathcal{I}$ (We also use $\bar{\mathcal{T}}$ and $\mathcal{T}_i$ to indicate a subset of $\mathcal{T}$, where $\bar{\mathcal{T}}$ is the keywords in a search request and $\mathcal{T}_i$ constitutes the ith level of $\mathcal{I}$, $i = 1, \ldots, h$).

2.2.2 Known Background Model

In this stronger model, the cloud server is equipped with more knowledge than what can be accessed in the known ciphertext model. In particular, the attacker may extract the statistical information from a known comparable dataset of the similar nature to the targeted dataset, e.g., the TF distribution information of a specific keyword. Given such statistical information, the cloud server is able to launch statistical attack to deduce/identify specific keywords in the query [9], [10], [11].

2.3 Design Goals

To enable effective, efficient and secure multi-keyword ranked search over encrypted cloud data under the aforementioned models, our mechanism is aiming to achieve the following design goals.

2.3.1 Accuracy-Improved Multi-Keyword Ranked Search

To design an encrypted cloud data search scheme which not only supports the effective *multi-keyword search*

functionality, but also, by adoption of the vector space model, achieves the accuracy-improved *similarity-based search result ranking*.

2.3.2 Search Efficiency

Instead of linear search [21], we explore a tree-based index structure and an efficient search algorithm to achieve better practical search efficiency.

2.3.3 Authenticity of Search Result

To make the proposed encrypted data search scheme verifiable and assure data user of authenticity of the returned search results.

2.3.4 Privacy Goals

The general goal is to protect user privacy by preventing the cloud server from learning information of the document set, the index tree, and the queries. In particular, search privacy requirements that we are concerned with are

1. *Index Confidentiality*: the underlying plaintext information pertaining to the encrypted index tree, e.g., keywords and TF of keywords;
2. *Query Confidentiality*: the plaintext information regarding the encrypted query, e.g., keywords in the query and document frequency (DF) of these keywords;
3. *Query Unlinkability*: whether two or more encrypted queries are from the same search request;
4. *Keyword Privacy*: the identification of specific keyword in the index tree, in the query or in the document set.

Note that protecting *access pattern*, i.e., the sequence of returned documents, is extremely expensive since the algorithm has to ''touch'' the whole document set [29]. We do not aim to protect it in this work for efficiency concerns.

2.4 Preliminaries

2.4.1 Vector Space Model

Among many similarity measures in plaintext information retrieval, vector space model [8] is the most popular one, supporting both conjunctive search and disjunctive search. Specifically, document rankings are realized by comparing the deviation of angles, i.e., cosine values, between each document vector and the query vector. The cosine measure allows accurate rankings due to the ''TF $\times$ IDF rule'', where TF denotes the occurrence count of a term within a document, and IDF is obtained by dividing the total number of documents in the collection by the number of documents containing the term. We adopt the similarity evaluation function for cosine measure $Cos(D_d, Q)$ from [8], where D_d is the index vector of document d for all the keywords in $\mathcal{T}$ and Q is the query vector for keyword set $\vec{\mathcal{T}}$. Both D_d and Q are unit vectors.

2.4.2 MD-Algorithm

The MD-algorithm [28] is used to find the k-best matches in a database that is structured as an MDB-tree [30], as shown in Fig. 2. In the database scenario, each level of the MDB-tree represents an attribute domain and each attribute in that domain is assigned an attribute value. All the attributes sharing the same value in the upper domain forms a child node. As such, a set of objects is allowed to be indexed in one data structure. An important search parameter, the prediction threshold value $\hat{P}_i$ for each level i, is obtained from the maximum attribute value P_i at each level, for example, in Fig. 2, $\hat{P}_i = P_i = 1.0$. Fig. 2 also shows an example that, when $k = 3$, the set of objects, E, K, and J, are returned to the user and the cross signs in the figure indicate that it is not necessary to access the nodes below.[1]

3 Secure Index Scheme

To achieve accurate multi-keyword ranked search, we adopt the cosine measure to evaluate similarity scores. In particular, we divide the original long document index vector D_d into multiple sub-vectors such that each sub-vector $D_{d,i}$ represents a subset of keywords $\mathcal{T}_i$ of $\mathcal{T}$, and becomes a part of the ith level of the index tree $\mathcal{I}$, as shown in Fig. 3. The query vector Q is divided in the same way as D_d is done. Let Q_i be the query sub-vector at the ith level. As such, the final similarity score for document d can be obtained by summing up the scores from each level. Based on these similarity scores, the cloud server determines the relevance of document d to the query Q and sends the top-k most relevant documents back to the user. By using the level-wise secure inner product scheme, the document index vector $D_{d,i}$ and the query vector Q_i are both well protected.

3.1 BMTS in Known Ciphertext Model

To facilitate the relevance rankings, the similarity scores, i.e., cosine values, are revealed to the cloud server, which differs from the schemes adopted in [21], [31]. In other words, we do not apply the dimension extension technique to our basic scheme in the known ciphertext model. For each level i of $\mathcal{I}$, BMTS scheme can be described as follows:

3.1.1 Setup

In this initialization phase, the secret key SK_i is produced by the data owner, including: 1) a $|\mathcal{T}_i|$-bit randomly generated vector S_i, where $|\mathcal{T}_i|$ is the length of $\mathcal{T}_i$; 2) two $(|\mathcal{T}_i| \times |\mathcal{T}_i|)$ invertible random matrices $\{M_{1,i}, M_{2,i}\}$. Hence, SK_i can be denoted as a 3-tuple $\{S_i, M_{1,i}, M_{2,i}\}$.

3.1.2 GenIndex $(\mathcal{DC}, SK_i)$

For each document d, the data owner generates an index vector $D_{d,i}$ according to $\mathcal{T}_i$, and each dimension is a normalized TF weight $w_{d,t}$. Next, the splitting procedure is applied to $D_{d,i}$, which splits $D_{d,i}$ into two random vectors as $\{D_{d,i}', D_{d,i}''\}$. Specifically, with the $|\mathcal{T}_i|$-bit vector S_i as a splitting indicator, if the jth bit of S_i is 0, $D_{d,i}'[j]$ and $D_{d,i}''[j]$ are set as the same as $D_{d,i}[j]$; if the jth bit of S_i is 1, $D_{d,i}'[j]$ and $D_{d,i}''[j]$ are set to two random numbers so that their

1. More details of the vector space model, the MD-algorithm and the MDB-tree can be found in the supplementary file of this paper, which is available in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.282 [8] and [28].

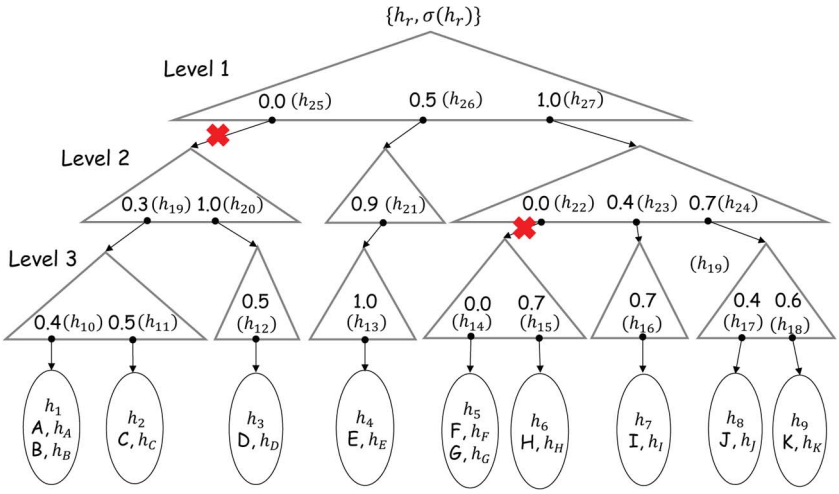


Fig. 2. Illustration of the MD-algorithm on the MDB-tree & Authentication for proposed secure index tree.

sum is equal to $D_{d,i}[j]$. Finally, the encrypted index vector $\widetilde{D_{d,i}}$ is built as $\{M_{1,i}^T D_{d,i}', M_{2,i}^T D_{d,i}''\}$.

3.1.3 GenQuery $(\bar{\mathcal{T}}, SK_i)$

With the keywords of interest in $\bar{\mathcal{T}}$, the query vector Q_i is generated, where each dimension is a normalized IDF weight $w_{q,t}$ ($w_{q,t} = 0$ for any keyword t not present in Q_i). Subsequently, Q_i is split into two random vectors as $\{Q_i', Q_i''\}$ with the similar splitting procedure. The difference is that if the jth bit of S_i is 0, $Q_i'[j]$ and $Q_i''[j]$ are set to two random numbers so that their sum is equal to $Q_i[j]$; if the jth bit of S_i is 1, $Q_i'[j]$ and $Q_i''[j]$ are set as the same as $Q_i[j]$. Finally, the encrypted query vector $\widetilde{Q_i}$ is yielded as $\{M_{1,i}^{-1} Q_i', M_{2,i}^{-1} Q_i''\}$.

3.1.4 SimEvaluation $(\widetilde{D_{d,i}}, \widetilde{Q_i})$

The cloud server executes similarity evaluation with query vector $\widetilde{Q_i}$ as in Eq. (1).

The similarity score at the ith level is:

$$\begin{aligned} Cos(\widetilde{D_{d,i}}, \widetilde{Q_i}) &= \left\{ M_{1,i}^T D_{d,i}', M_{2,i}^T D_{d,i}'' \right\} \\ &\quad \cdot \left\{ M_{1,i}^{-1} Q_i', M_{2,i}^{-1} Q_i'' \right\} \\ &= D_{d,i}' \cdot Q_i' + D_{d,i}'' \cdot Q_i'' \\ &= D_{d,i} \cdot Q_i. \end{aligned} \tag{1}$$

Hence, the final similarity score for document d is $\sum_{i=1}^{h} D_{d,i} \cdot Q_i = D_d \cdot Q$.

3.1.5 Security Analysis

We analyze BMTS with respect to the search privacy requirements as described in Section 2.

1. **Index confidentiality and Query confidentiality**: In BMTS, $\widetilde{D_{d,i}}$ and $\widetilde{Q_i}$ are obfuscated vectors. As long as the secret key SK_i is kept confidential, the cloud server cannot infer the original vectors $D_{d,i}$ or Q_i. Neither can it deduce the keywords nor the TF and IDF information included in the documents or queries from the result similarity scores, which appear to be random values to the server. This has been proven in the known ciphertext model in [31].
2. **Query unlinkability**: The adopted vector encryption method provides non-deterministic encryption, in light of the random vector splitting procedure. Thus the same search keywords will be encrypted to different query vector $\widetilde{Q}$. The non-linkability of search requests can be provided to this extent. However, if a cloud server is capable of tracking the nodes visited and the intermediate similarity results, it is possible for the cloud server to link the same search request based on the same similarity scores. In this case the *search pattern* or the *access*

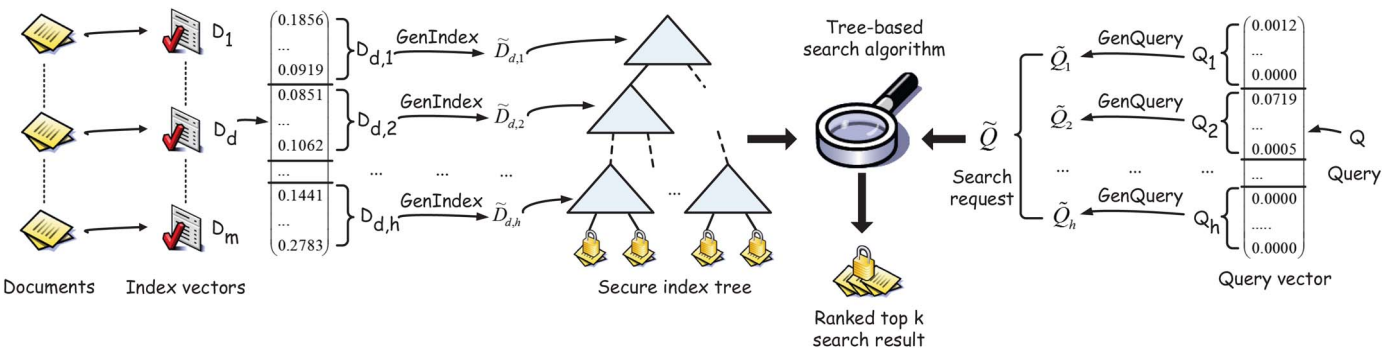


Fig. 3. Overview of secure index scheme.

pattern will be leaked even in the known ciphertext model.

3. **Keyword privacy**: In the known background, the highly motivated attacker can identify a particular keyword by exploiting the distribution of similarity score of this keyword (The detail explanation can be found in the supplementary file of this paper available online).

To enhance security and boost search efficiency, search evaluation may be only executed at certain levels where the user-intended keywords reside; for the other levels, we can render these similarity scores some fixed values, e.g., 0, during the execution of both similarity score prediction and evaluation in the search process.

3.2 EMTS in Known Background Model

The previous security analysis shows that in the known background model, *keyword privacy* breach is possible, due to the distance-preserving property of BMTS, i.e., the cosine value calculated from $\widetilde{D_{d,i}}$ and $\widetilde{Q_i}$ is equal to the one from $D_{d,i}$ and Q_i. To break such equality, we introduce some tunable randomness into the similarity score evaluation, by which the cloud server cannot differentiate keywords from the particular similarity score distributions. In addition, this randomness can be calibrated by the user to represent the user's preference for more accurate ranked search result versus better-protected *keyword privacy*. Specifically, U_i phantom terms are added into the query vector Q_i, and we extend the index vector $D_{d,i}$ from $|\mathcal{T}_i|$ dimensions to $|\mathcal{T}_i| + U_i$ dimensions. We denote the subset of h levels where the keywords of interest reside as w and its size $|w| \leq h$.

Our EMTS scheme is performed almost the same as BMTS, except that at the ith level: 1) in Setup phase, S_i becomes $(|\mathcal{T}_i| + U_i)$-bit long. $M_{1,i}$ and $M_{2,i}$ are $(|\mathcal{T}_i| + U_i) \times (|\mathcal{T}_i| + U_i)$ dimensional matrices; 2) in GenIndex phase, by choosing V_i out of U_i phantom terms, the corresponding entries in the $(|\mathcal{T}_i| + U_i)$-dimensional index vector $D_{d,i}$ are set to 1; 3) when generating encrypted query, the $(|\mathcal{T}_i| + j)$th entry in Q_i where $j \in [1, U_i]$ is set to a random number $\varepsilon_{i,j}$; 4) The cloud server executes similarity evaluation and obtains the final similarity score for document d equal to $(D_d \cdot Q + \sum_{i \in w} \sum_{j \in \bar{V}_i} \varepsilon_{i,j})$, where $\bar{V}_i$ is the set of the V_i selected phantom terms, and it is different for each index vector at level i.

3.2.1 Security Analysis

We analyze EMTS again regarding the aforementioned search privacy requirements.

1. **Index confidentiality and Query confidentiality**: EMTS can protect *index* and *query confidentiality* in both the known ciphertext model and the known background model, which is inherited from BMTS.
2. **Query unlinkability**: The introduction of randomly generated $\varepsilon_{i,j}$ will allow EMTS to produce different similarity scores even for the same search request. The value of $\varepsilon_{i,j}$ can be adjusted to control the level of variance thus the level of unlinkability. It is worth noting that this query-side randomization technique significantly differs from [21], where randomization occurs on the index vector side and is not possible to be tweaked as an effective privacy-preserving parameter for users. Query unlinkability is thus much enhanced compared with BMTS to the extent that there is no easy way for the attacker to link the queries. However, since we do not intend to protect access pattern for efficiency reasons, the returned results from the same request will always bear some similarity which could be exploited with powerful statistical analysis by the very motivated cloud server. This is a trade-off that one has to make between efficiency and privacy.
3. **Keyword privacy**: By carefully selecting phantom terms in the query, the final similarity scores in EMTS are obfuscated such that the corresponding distribution is not keyword specific any more, from which the attacker is not able to reverse-engineer the underlying keyword as in BMTS (See the more detailed discussion in the supplementary file available online).

Remarks. Recently Yao *et al.* [32] found that this underlying encryption method [31] was susceptible to chosen plaintext attack. However, it is not applicable under our defined threat models, since to launch such attack, the cloud server has to acquire the vector representation of the query, which are *only* possessed by the data owner and protected by BMTS and EMTS.

4 Efficiency of the Tree-Based Search Algorithm

In the plaintext information retrieval community, many well-developed techniques have been adopted to accelerate the search process, e.g., inverted index [33], B-tree [34], etc. However, in the ciphertext scenario, they cannot be implemented in a straightforward manner. In [9], [10], [11], [19], the inverted index based search methods are employed to achieve an extremely efficient search process. However, these schemes are only designed for single keyword search. Efficient range search in database [23] can be realized by using B^+-tree, but it is not applicable to the text search scenario. The similarity score in our scheme is a value depending on the query and has to be evaluated in the runtime, which makes the fixed tree structures, such as B-tree or B^+-tree, not suitable here. In this paper, we propose a tree-based search algorithm, which is adapted from MDB-tree based MD-algorithm, to enable efficient multi-keyword ranked search. In what follows, we briefly introduce our tree-based search algorithm and present some experimental results from our implementation of the proposed tree-based search algorithm on a real-world document set: the recent ten years' INFOCOM publications. We identify key factors that affect the search efficiency and propose strategies in building the index tree that effectively speed up the search process.

4.1 Tree-Based Search Algorithm

The MD-algorithm is originally designed for plaintext database search. In the case of privacy-preserving similarity-based multi-keyword ranked text search, it cannot be applied

in a straightforward manner. Instead of a numerical ''attribute value'' for each attribute in the MDB-tree, our index tree structure has to be built on vectors. The secure index scheme described in Section 3 is for this purpose and it enables the search algorithm to take the inputs of the encrypted searchable index tree and the encrypted query, and ensures that the search algorithm is conducted in a secure way to protect important search privacy in the whole search process.

Another remarkable difference between our search algorithm and the MD-algorithm is that we cannot set $\hat{P}_i$ to P_i as running the MD-algorithm in database scenario, since P_i varies for queries in our scenario and has to be securely evaluated (as described in Section 3) in the runtime.

4.2 Impact of Prediction Threshold Value

An important factor that affects the search efficiency is the prediction threshold value $\hat{P}_i$ at each level i. To ensure the search precision, $\hat{P}_i \geq P_i$ should hold where P_i is the maximum similarity score at level i. The tighter the prediction value of $\hat{P}_i$, the higher the search efficiency. The reason is that the search process can be terminated earlier without going into unnecessary nodes. On the other hand, when $\hat{P}_i < P_i$, the search precision (a quantitative measure for search accuracy, cf. Section 6) drops while the rank privacy (a privacy measure, cf. Section 6) increases.

4.2.1 Strategy 1

Based on this observation, our first efficiency enhancement aims to produce a better estimation of $\hat{P}_i$ that approximates to its ideal value P_i. We propose the following strategy to achieve this. During the index tree generation phase, the data owner retains a vector E_i for each level i. This vector consists of the maximum values at each dimension among all the indexes at this level. Subsequently, during the query generation phase, $\hat{P}_i$ is equal to the inner product of E_i and Q_i, and $\hat{P}_i$ will be set to 1 if it is greater than 1, thus $P_i \leq \hat{P}_i \leq 1$. $\hat{P}_i$ can be taken as an additional search parameter to be sent with $\widetilde{Q_i}$ to the cloud server. As for EMTS, we add the maximum $\sum_{j \in \bar{V}_i} \varepsilon_{i,j}$ from Q_i to $\hat{P}_i$, and refer to this sum as the final prediction threshold value.

4.3 Impact of Intended Keyword Position

Another factor we observed that affects the search efficiency is the position of the search keywords on the index tree. The higher level the intended keywords reside, the higher the search efficiency. This is very different from using the MD-algorithm in database scenario where all the attributes are involved in searching the relevant objects. In the text search scenario, people are likely to complete a search with a query only comprising five keywords or less [35]. Consequently, the search algorithm needs to go through a larger number of nodes to evaluate an intended keyword if it resides at a lower level.

4.3.1 Strategy 2

The insight from this observation is that the average search time can be improved by strategically arranging keyword position in the index tree, i.e., the most frequently searched keywords on the top levels. In practice, the information on the search keyword distribution can be extracted from the user search history.

4.4 Impact of Index Vector Clustering

Another idea for improving the search efficiency is to cluster ''similar'' index vectors. The improved efficiency comes from the reduced number of accessed nodes in the index tree, but at the expense of lower search precision. The bigger each cluster is, the higher the search efficiency, but the lower the search precision.

4.4.1 Strategy 3

To maximize the possibility of clustering, the length of the index vector at each level should be as short as possible (but at least achieve 80-bit symmetric key security [31]) to group the ''similar'' indexes. Inspired by the k-means method, which is the most widely used clustering technique in the data mining community [36], we use Euclidean distance (Ed) as a metric to cluster ''close enough'' vectors, e.g., when $Ed < 1$. For EMTS, we may first cluster original index vectors, and then execute dimension extension.

Remarks. The original combination of the MD-algorithm and the MDB-tree is not directly applicable for efficient search over vector indexes. From the extensive experiments on our prototype implementation of the search algorithm (cf. the supplementary file for detailed experimental result available online), we identified three efficiency-crucial factors and proposed effective strategies to improve the practical search efficiency with our vector indexes. Although the worst case search complexity is no better than linear search that is the state-of-the-art search efficiency in the multi-keyword encrypted text search scenario, this much less time-consuming tree-based search algorithm represents a solid step forward on the utilization of encrypted cloud data in practice. From the security point of view, the entire search process does not introduce new privacy vulnerability when used with our secure index scheme. In particular, our scheme is secure against search time analysis, i.e., the cloud server cannot infer specific keyword by the difference of search time, even if he/she knows that the keyword resides at a certain level. In fact, for efficiency, the cloud server performs level-selected similarity evaluation (see Section 3), such that he/she has already possessed the keyword position information. There are at least 80 dimensions in an index vector at each level, and all the words falling into this level have almost the same search time. This effectively blinds one keyword within at least 79 other keywords at the same level. In addition, the cloud server has no knowledge about which concrete set of keywords are selected to build this level without the dictionary $\mathcal{T}$. Therefore, it is not possible to differentiate these keywords or identify a particular keyword of interest.

5 Search Result Verification

Due to possible data corruption, software malfunction, and intention to save computational resources by the cloud server, etc., search result returned to the user may be false

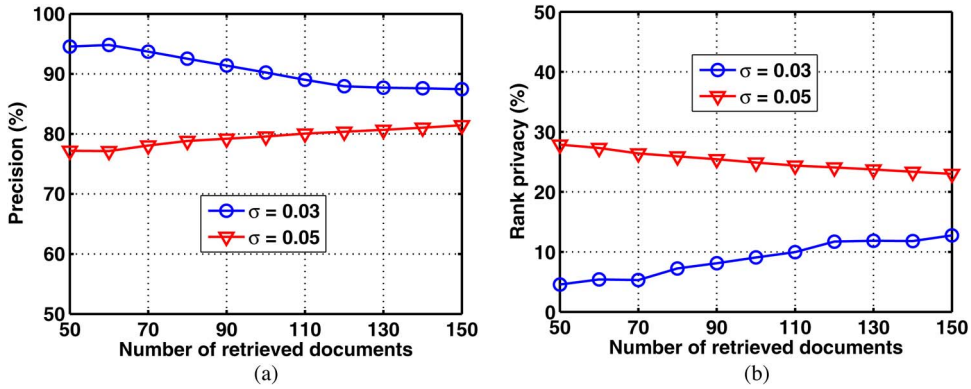


Fig. 4. By choosing different standard deviation σ, the trade-off, between (a) Precision, and (b) Rank privacy, can be achieved.

or contain errors. A mechanism to verify the authenticity of the search results by the user is desirable. Thus in this section we aim to design an authenticity check mechanism over the returned search results to address this problem. First, we have the following definition.

Definition. *The authenticity of the returned search results includes three aspects: 1) Correctness: the returned search results do exist in the dataset and remain unmodified; 2) Completeness: no qualified documents are omitted from the search results; 3) Freshness: the returned results are acquired from the latest version of the dataset.*

Our design should allow the user to verify the above properties. The main idea behind our scheme is to let cloud server return the minimum sub-tree of the proposed secure index tree. Then data user searches this minimum tree using the same search algorithm as the cloud server did, which suffices to assure user of the authenticity of the query results. However, to realize this design goal, we must be able to have the returned minimum index tree authenticated first. After that, by using the proposed search algorithm, we could achieve the defined objectives readily. Data owner could turn the existing secure index tree structure into an authenticated one through cryptographic signature σ (e.g., RSA signature) and collision-resistant hash function h (e.g., SHA-2) as follows.

At the leaf node, the data owner computes the hash values h_{ID_i} of each document ID_i within this node in the forms of $\{h(ID_i \| \Phi(ID_i))\}_{1 \le i \le z}$, where $\|$ represents concatenation, $\Phi(ID_i)$ denotes the content of the document ID_i and z is the total number of documents in this leaf node. Then the data owner could generate the hash value of this node as $h(h_{ID_1} \| \ldots \| h_{ID_z})$. For example, in Fig. 2, the leaf node hash value h_1 can be generated from documents A, B and thus is defined to be $h(h_A \| h_B)$. For each non-leaf node at the ith level, every index $\widetilde{D_{d,i}}$ is with its hash value $h(\widetilde{D_{d,i}} \| h_{child})$, where h_{child} is the hash value of its child node. Let $h(\widetilde{D_{d,i}}) = h(e_1 \| \ldots \| e_{n_i})$, in which e is the coordinate of the vector index $\widetilde{D_{d,i}}$ and n_i is its dimension. The hash value of this non-leaf node can be generated by hashing all the index hash values within. For instance, in Fig. 2, $h_{10} = h(\widetilde{D_{A,3}} \| h_1)$ and the hash value of this node is $h(h_{10} \| h_{11})$. Similarly, the data owner hashes all the index hash values inside the root node to obtain h_r, e.g., $h_r = h(h_{25} \| h_{26} \| h_{27})$ in Fig. 2. Then the data owner signs this hash value $\sigma(h_r)$ and outsources it to the cloud server.

To execute the query result verification, the cloud server sends back the final ranked search results along with minimum secure index tree, and the corresponding auxiliary information aux including the hash values of the unreturned documents in the leaf nodes in this minimum tree, the hash values of the necessary non-leaf nodes outside the sub-tree, which are not accessed in the search process, and the root signature $\sigma(h_r)$. Given these information, the data user can validate the tree structure by verifying the signature $\sigma(h_r)$, i.e., the order of the nodes (leaf and non-leaf nodes), the order and integrity of the vector indexes in the non-leaf nodes, and the order and integrity of documents in the leaf nodes. Take Fig. 2 for example, the requested documents are E, K, and J. The minimum index tree returned is also shown in Fig. 2 but without the structures below the cross signs. Set the auxiliary information aux to be $\{h_I, h(h_{19} \| h_{20}), h(h_{14} \| h_{15})\}, \sigma(h_r)$. As such, the data user can compute h_r easily and verify the root signature. Now the data user can search this minimum secure index tree with the encrypted query $\widetilde{Q}$ and be convinced that all the intended documents with the keywords of interest have been returned.

Remarks. Only data owner can issue the valid root signature $\sigma(h_r)$. Thus, any data corruption would lead to a mismatch with the signed root of the secure index tree. By verifying $\sigma(h_r)$, data user can be assured of the existence of the returned documents in the dataset and their integrity. After searching the verified sub-tree by using the same search algorithm, data user is able to check whether the ranking order of the returned documents is correct and all the qualified documents are retrieved. As a result, the proposed scheme can achieve the defined objectives of *Correctness* and *Completeness*. For *Freshness*, the data owner could add time stamp into the root signature as $\sigma(h_r \| ts)$, where ts indicates the date of index update last time. Data user can obtain this information from the data owner along with the encrypted query. As such, we make the whole secure search system verifiable and the proposed search result verification scheme satisfies our design goal.

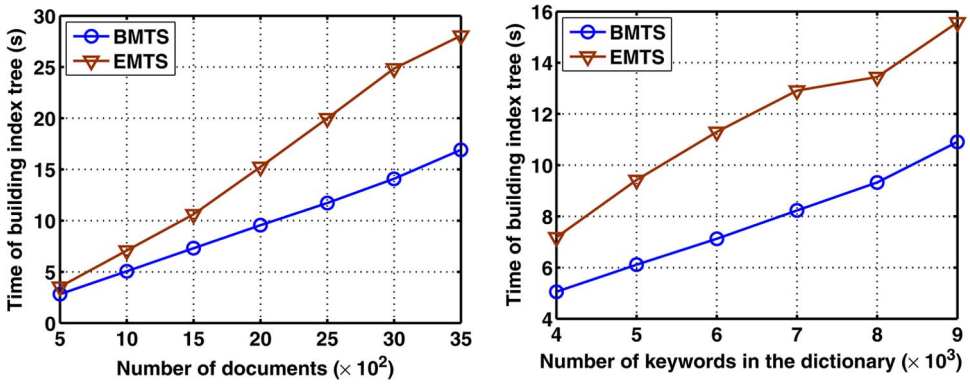


Fig. 5. Time cost for building index tree. (a) For the different size of document set with the same dictionary, $n = 4000$. (b) For the different size of dictionary with the same document set, $m = 1000$.

6 Performance Evaluation

To evaluate the overall performance of our proposed techniques, we implemented the entire secure search system using JAVA on a Linux Server with Intel Core i3 Processor 3.3 GHz. The document set is built from the recent ten years' IEEE INFOCOM publications, including about 3600 publications, from which we extract about 9000 keywords. In this section we present the detailed performance result. The documents and keywords used in the evaluation are selected randomly from the created document sets.

1. **Precision and Privacy:** To evaluate the impact on the accuracy of search result introduced by phantom terms in EMTS, we adopt the definition of ''precision'' in [21]. Namely, the ''precision'' of a top-k search is defined as $P_k = k'/k$ where k' is the number of the real top-k documents that are returned by the cloud server. Fig. 4a shows that with a small σ, the effectiveness of the search scheme is not affected much. The user can still enjoy almost the same search result as BMTS. On the other hand, we evaluate the ''rank privacy'' obtained from introducing phantom terms, whose definition is also adopted from [21], i.e., the rank privacy at point k is calculated as $\widetilde{P_k} = \sum \widetilde{p_k}/k^2$. For every document d in the returned top-k documents, let the rank perturbation $\widetilde{p_k}$ be $|u_d - u_d'|$, where u_d is the rank number of document d in the returned top-k documents and it is set to k if greater than k, and u_d' is its rank number in the real ranked documents. As shown in Fig. 4b, large σ provides better protection of rank information in EMTS. It is worth noting that σ is a tunable search parameter at the discretion of the user. The selection of different σ reflects his/her predilection for the better effectiveness of the search scheme or the better protected rank privacy and *keyword privacy* (see Section 3.2).
2. **Construction for Index Tree:** Fig. 5a shows that the time cost for building the index tree is nearly linear to the number of the documents, given the same dictionary. Fig. 5b shows that with the same document set, the index construction time is proportional to the number of keywords in the dictionary. On the other hand, considering the massive storage capacity and low storage cost in the cloud, the storage overhead is practical and completely affordable (cf. the supplementary file of this paper available online).
3. **Query Generation:** Fig. 6a demonstrates that when $|\mathcal{T}_i|$ is fixed, the time cost for generating an encrypted query is only linear to the number of levels where the searched keywords reside. Since we may place the

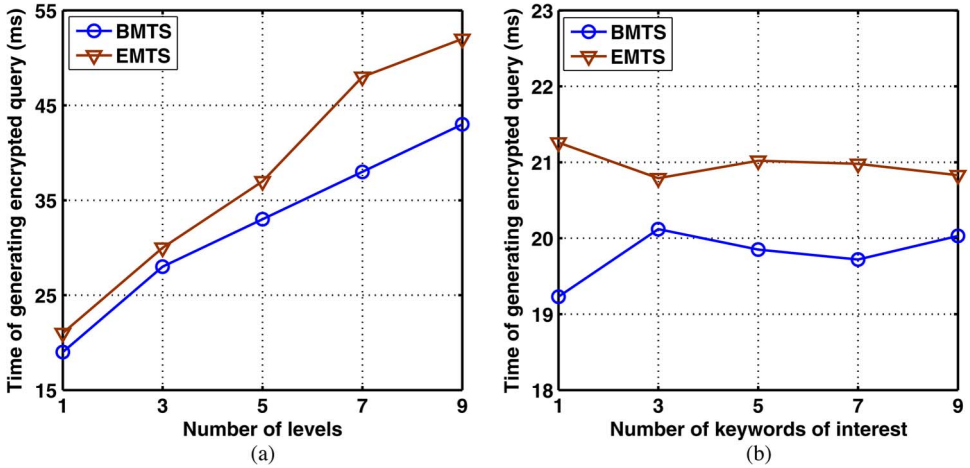


Fig. 6. Time cost for generating encrypted query, when $|\mathcal{T}_i| = 80$. (a) For the different number of levels where user intended keywords reside. (b) For the different number of keywords of interest in one level.

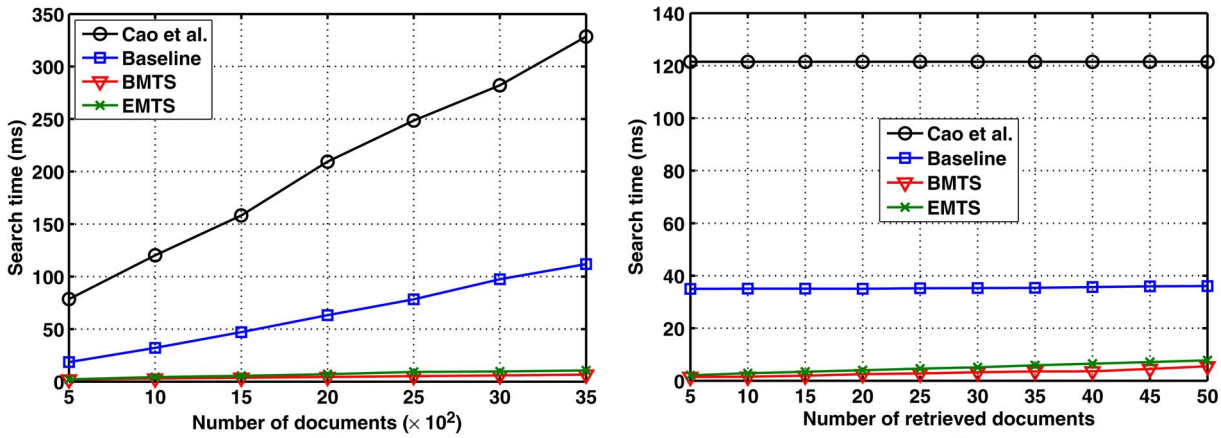


Fig. 7. Search efficiency, with the same 10 keywords. (a) For the different size of document set with the same dictionary, $n = 4000$, $k = 10$. (b) For the different number of retrieved documents with the same document set and dictionary, $m = 1000$, $n = 4000$.

most frequently searched keywords on the top levels of $\mathcal{I}$, a good portion of queries are only generated for a few limited levels. As such, when $|\mathcal{T}_i|$ is chosen properly, the average query generation can be extremely efficient, as shown in Fig. 6b.

4. **Search Efficiency:** The time cost of our proposed encrypted cloud data search is much more efficient than [21] and baseline search (cf. the supplementary file of this paper available online) as shown in Fig. 7a. In addition, with the increased size of document set, our two proposed schemes enjoy almost the same and nearly constant search time. Fig. 7b demonstrates that when user requests more relevant documents, our search algorithm is still extremely efficient.

7 Conclusion

In this paper, we first exploit the popular similarity measure, i.e., vector space model with cosine measure, to effectively procure the accurate search result. We propose two secure index schemes to meet various privacy requirements in the two threat models. Eventually, the leakage of sensitive frequency information can be avoided. To boost search efficiency, we propose a tree-based index structure for the whole document set. From the utilization of the prototype of our secure search system, we identify three essential efficiency-related factors, by which the efficiency of the search algorithm upon our index tree can be significantly improved. In addition, we make the whole search process verifiable in case that users want to ensure the authenticity of the returned search results. Finally, thorough evaluation on the real-world document set demonstrates the performance of BMTS and EMTS in terms of search effectiveness, efficiency and privacy.

Acknowledgment

The authors would like to thank M. Ondreička for his helpful comments and sharing the source code of MD-algorithm. This work was supported in part by the NSFC 61272457, the FRFCU K50511010001, the PCSIRT 1078, the National 111 Project B08038 and the U.S. NSF grants CNS-1217889, CNS-1155988, and CNS-1218085. A preliminary version [1] of this paper was presented at the 8th ACM Symposium on Information, Computer and Communications Security (ACM ASIACCS'13).

References

[1] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, and H. Li, ''Privacy-Preserving Multi-Keyword Text Search in the Cloud Supporting Similarity-Based Ranking,'' in *Proc. ACM ASIACCS*, 2013, pp. 71-82.

[2] Cloud Security AllianceSecurity Guidance for Critical Areas of Focus in Cloud Computing V3.0. [Online]. Available: http://www.cloudsecurityalliance.org 2011

[3] J. Sheridan and C. Cooper, *Defending the Cloud*. [Online]. Available: http://www.reactionpenetrationtesting.co.uk/Defending

[4] Z. Slocum, *Your Google Docs: Soon in Search Results?* [Online]. Available: http://www.cnet.com/news/your-google-docs-soon-in-search-results/

[5] B. Krebs, *Payment Processor Breach May Be Largest Ever*. 2009. [Online]. Available: http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html

[6] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, ''Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131-143, Jan. 2013.

[7] S. Yu, C. Wang, K. Ren, and W. Lou, ''Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing,'' in *Proc. IEEE INFOCOM*, 2010, pp. 1-9.

[8] I.H. Witten, A. Moffat, and T.C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*. San Francisco, CA, USA: Morgan Kaufmann, May 1999.

[9] 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.

[10] A. Swaminathan, Y. Mao, G.-M. Su, H. Gou, A.L. Varna, S. He, M. Wu, and D.W. Oard, ''Confidentiality-Preserving Rank-Ordered Search,'' in *Proc. ACM Workshop Storage Security Survivability*, 2007, pp. 7-12.

[11] S. Zerr, D. Olmedilla, W. Nejdl, and W. Siberski, ''Zerber+r: Top-k Retrieval from a Confidential Index,'' in *Proc. EDBT*, 2009, pp. 439-449.

[12] D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano, ''Public Key Encryption with Keyword Search,'' in *Proc. EUROCRYPT*, 2004, pp. 506-522.

[13] P. Golle, J. Staddon, and B.R. Waters, ''Secure Conjunctive Keyword Search Over Encrypted Data,'' in *Proc. ACNS*, 2004, pp. 31-45.

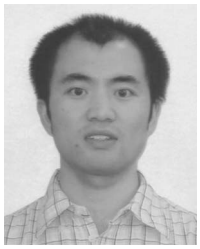
[14] D. Boneh and B. Waters, ''Conjunctive, Subset, and Range Queries on Encrypted Data,'' in *Proc. TCC*, 2007, pp. 535-554.

[15] Y. Hwang and P. Lee, ''Public Key Encryption with Conjunctive Keyword Search and its Extension to a Multi-User System,'' in *Proc. Pairing*, 2007, pp. 2-22.

[16] D. Song, D. Wagner, and A. Perrig, ''Practical Techniques for Searches on Encrypted Data,'' in *Proc. IEEE S P*, 2000, pp. 44-55.
[17] E.-J. Goh, ''Secure Indexes,'' in *Cryptology ePrint Archive* 2003. [Online]. Available: http://eprint.iacr.org/2003/216
[18] Y.-C. Chang and M. Mitzenmacher, ''Privacy Preserving Keyword Searches on Remote Encrypted Data,'' in *Proc. ACNS*, 2005, pp. 391-421.
[19] R. Curtmola, J.A. Garay, S. Kamara, and R. Ostrovsky, ''Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions,'' in *Proc. ACM CCS*, 2006, pp. 79-88.
[20] P. Liesdonk, S. Sedghi, J. Doumen, P. Hartel, and W. Jonker, ''Computationally Efficient Searchable Symmetric Encryption,'' in *Proc. Secure Data Manage.*, 2010, pp. 87-100.
[21] 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*, 2011, pp. 829-837.
[22] E. Shen, E. Shi, and B. Waters, ''Predicate Privacy in Encryption Systems,'' in *Proc. TCC*, 2009, pp. 457-473.
[23] Y. Lu, ''Privacy-Preserving Logarithmic-Time Search on Encrypted Data In Cloud,'' in *Proc. NDSS*, 2012, pp. 1-17.
[24] H. Pang, J. Shen, and R. Krishnan, ''Privacy-Preserving Similarity-Based Text Retrieval,'' *ACM Trans. Internet Technol.*, vol. 10, no. 1, p. 4, Feb. 2010.
[25] F. Li, M. Hadjieleftheriou, G. Kollios, and L. Reyzin, ''Dynamic Authenticated Index Structures for Outsourced Databases,'' in *Proc. SIGMOD*, 2013, pp. 121-132.
[26] H. Pang and K.-L. Tan, ''Authenticating Query Results in Edge Computing,'' in *Proc. ICDE*, 2004, pp. 560-571.
[27] H. Pang and K. Mouratidis, ''Authenticating the Query Results of Text Search Engines,'' *Proc. VLDB Endowment*, vol. 1, no. 1, pp. 126-137, Aug. 2008.
[28] M. Ondreička and J. Pokorný, ''Extending Fagin's Algorithm for More Users Based on Multidimensional B-Tree,'' in *Proc. ADBIS*, 2008, pp. 199-214.
[29] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, ''Private Information Retrieval,'' *J. ACM*, vol. 45, no. 6, pp. 965-981, Nov. 1998.
[30] P. Scheuermann and M. Ouksel, ''Multidimensional B-Trees for Associative Searching in Database Systems,'' *Inf. Syst.*, vol. 7, no. 2, pp. 123-137, 1982.
[31] W.K. Wong, D.W. Cheung, B. Kao, and N. Mamoulis, ''Secure KNN Computation on Encrypted Databases,'' in *Proc. SIGMOD*, 2009, pp. 139-152.
[32] B. Yao, F. Li, and X. Xiao, ''Secure Nearest Neighbor Revisited,'' in *Proc. ICDE*, 2013, pp. 733-744.
[33] NIST, Gaithersburg, MD, USANIST's Dictionary of Algorithms and Data Structures: Inverted Index. [Online]. Available: http://xlinux.nist.gov/dads//HTML/invertedIndex.html
[34] D. Comer, ''Ubiquitous B-Tree,'' *ACM Comput. Surveys*, vol. 11, no. 2, pp. 121-137, June 1979.
[35] Keyword and Search Engines Statistics. 2013. [Online]. Available: http://www.keyworddiscovery.com/keyword-stats.html?date=2013-01-01
[36] A. Rajaraman and J.D. Ullman, *Mining of Massive Datasets*. Cambridge, U.K.: Cambridge Univ. Press, Dec. 2011.

Wenhai Sun received the BS degree in information security from Xidian University, Xi'an, China, in 2007. Since 2009, he has been a PhD student in a combined MS/PhD program in the School of Telecommunications Engineering at Xidian University. From 2011 to 2013, he was a visiting PhD student in the Cyber Security Lab at Virginia Tech. His research interests are applied cryptography, cloud computing security and wireless network security. He is a Student Member of the IEEE.

Bing Wang received the BS degree from Fudan University and the ME degree from Shanghai Jiaotong University, both in computer science, in 2008 and 2011, respectively. He is currently pursuing the PhD degree in the Computer Science Department at Virginia Tech. His research interests are in the areas of applied cryptography and network security, with current focus on secure data service outsourcing in cloud computing. He is a Student Member of the IEEE.



Ning Cao received the BE and ME degrees in Computer Science from Xi'an Jiaotong University in China, and the PhD degree in electrical and computer engineering from Worcester Polytechnic Institute. He joined the Research and System Infrastructure at Google Inc. in 2012. His research interests are in the areas of security, privacy, and reliability in cloud computing, with current focus on search and storage. He is a member of ACM. He is a member of the IEEE.

Ming Li (S'08-M'11) received the BE and ME in degrees in electronic and information engineering from Beihang University, China, and the PhD degree in electrical and computer engineering from Worcester Polytechnic Institute. In 2011, he joined the Computer Science Department at Utah State University as an Assistant Professor. His research interests are in the general areas of cyber security and privacy, with current emphases on data security and privacy in cloud computing, security in wireless networks and cyber-physical systems. He is a member of ACM. He is a member of the IEEE.

Wenjing Lou (M'03-SM'08) received the PhD in electrical and computer engineering at the University of Florida in 2003. She is an Associate Professor at Virginia Polytechnic Institute and State University. Prior to joining Virginia Tech in 2011, she was a faculty member at Worcester Polytechnic Institute from 2003 to 2011. Her current research interests are in cyber security, with emphases on wireless network security and data security and privacy in cloud computing. She was a recipient of the U.S. National Science Foundation CAREER award in 2008. She is a Senior Member of the IEEE.

Y. Thomas Hou (S'91-M'98-SM'04-F'14) is a Professor in the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA. His research interests are cross-layer optimization for wireless networks. He is also interested in wireless security. He has published extensively in leading journals and top-tier conferences and received five best paper awards from IEEE (including IEEE INFOCOM 2008 Best Paper Award and IEEE ICNP 2002 Best Paper Award) and one Distinguished Paper Award from ACM. Prof. Hou is currently serving as an Area Editor of *IEEE Transactions on Wireless Communications*, an Associate Editor of *IEEE Transactions on Mobile Computing*, an Editor of *IEEE Journal on Selected Areas in Communications* (Cognitive Radio Series), and an Editor of *IEEE Wireless Communications*. He is the Chair of IEEE INFOCOM Steering Committee. He is a Fellow of the IEEE.

Hui Li (M'10) received BSc degree from Fudan University in 1990 and the MSc and PhD degrees from Xidian University in 1993 and 1998. In 2009, he was with Department of Electrical and Computer Engineering, University of Waterloo as a visiting scholar. Since 2005, he has been a Professor in the School of Telecommunications Engineering, Xidian University, China. His research interests are in the areas of cryptography, security of cloud computing, wireless network security and information theory. He served as TPC co-chair of ISPEC 2009 and IAS 2009, general co-chair of E-Forensic 2010, ProvSec 2011 and ISC 2011. He is a member of the IEEE.