Privacy-Preserving Keyword Search Over Encrypted Data in Cloud Computing

Wenhai Sun, Wenjing Lou, Y. Thomas Hou, and Hui Li

Abstract Search over encrypted data is a technique of great interest in the cloud computing era, because many believe that sensitive data has to be encrypted before outsourcing to the cloud servers in order to ensure user data privacy. Devising an efficient and secure search scheme over encrypted data involves techniques from multiple domains – information retrieval for index representation, algorithms for search efficiency, and proper design of cryptographic protocols to ensure the security and privacy of the overall system. This chapter provides a basic introduction to the problem definition, system model, and reviews the state-of-the-art mechanisms for implementing privacy-preserving keyword search over encrypted data. We also present one integrated solution, which hopefully offer more insights into this important problem.

1 Introduction

We are in such an information-explosion era that constantly purchasing new hardware, software and training IT personnel is becoming a nightmare for almost every IT practitioner. Fortunately, we are witnessing an enterprise IT architecture shift

W. Sun (✉)
The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi, China

Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
e-mail: whsun@xidian.edu.cn

W. Lou • Y.T. Hou
Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
e-mail: wjlou@vt.edu; thou@vt.edu

H. Li
The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, Shaanxi, China
e-mail: lihui@mail.xidian.edu.cn

S. Jajodia et al. (eds.), *Secure Cloud Computing*, DOI 10.1007/978-1-4614-9278-8_9,

to a centralized, more powerful computing paradigm – Cloud Computing, in which enterprise's or individual's databases and applications are moved to the servers in the large data centers (i.e. the cloud) managed by the third-party cloud service providers (CSPs) in the Internet. Cloud computing has been gradually recognized as the most significant turning point in the development of information technology during the past few years. People are fascinated by the benefits it offers, such as ubiquitous and flexible access, on-demand computing resources configuration, considerable capital expenditure savings, etc. Indeed, many companies, organizations, and individual users have adopted the cloud platform to facilitate their business operations, research, or everyday needs [35].

Despite the tremendous business and technical advantages, what we shall always keep in mind is that cloud computing would not be our wonderland until users' outsourced sensitive data could hide from the prying eyes. Privacy concern is one of the primary hurdles that prevent the widespread adoption of the cloud by potential users, especially if the private data of users used to reside in the local storage are to be outsourced to and computed in the cloud. Imagine that CSPs host the services looking into your personal emails, financial and medical records, and social network profiles. Although these sensitive data could be protected by deploying intrusion detection systems, firewalls, or even segmenting data in a virtualized environment, CSP possesses full control of the cloud infrastructure including the system hardware and lower levels of software stack. Privacy breach is still likely to occur owing to the existence of disgruntled, profiteered or curious employees from CSP [25, 37]. Encrypting-then-outsourcing [28,48] provides us strong guarantee that no one could mine any useful information from the ciphertext of users' data. Many people argue that sensitive data has to be encrypted before outsourcing in order to provide user data privacy against the cloud service providers. However, encrypted data makes data utilization a very challenging task. One example is keyword search functions on the documents stored in the cloud. Without those usable data services, the cloud will become merely a remote storage which provides limited value to all parties.

Computation over encrypted data is a challenging task and has drawn significant attention due to the encrypting-then-outsourcing paradigm in cloud computing. It will be remiss if we don't mention fully homomorphic encryption [16], which is considered the Holy Grail of cryptography. Fully homomorphic encryption scheme will allow us to operate directly over ciphertext and generate results matching the computation over plaintext. A theoretical break-through on fully homomorphic encryption took place a few years ago [16]. However, the efficiency of the construction is still far from being practical. Much research work has been focusing on special classes of computation [2, 3, 19, 44]. Search over encrypted data is a fundamental and common form of data utilization service, enabling users to quickly sort out information of interest from huge amount of data, and thus has become a topic of great interest recently. Both public key cryptography (PKC) and symmetric key cryptography (SKC) can be used to build encrypted data search schemes. Generally speaking, PKC-based schemes [7,9,18,20] are more expressive, support more flexible search functions, but more computationally intensive, while

SKC-based schemes [11, 15, 17, 42] are more efficient in searching, but less flexible in the types of search criteria supported.

This chapter aims to provide a general overview of search techniques over encrypted data and their security and privacy objectives, and then elaborate on a scheme that can achieve privacy-preserving multi-keyword search supporting similarity-based ranking, based on [10] and [39]. The chapter is organized as follows. In Sect. 2, we will introduce the encrypted data search problem in terms of its problem formulation and review related works. We will delve into multi-keyword ranked search in Sect. 3, and further improve search result accuracy and search efficiency in Sect. 4. We will conclude this chapter in Sect. 5.

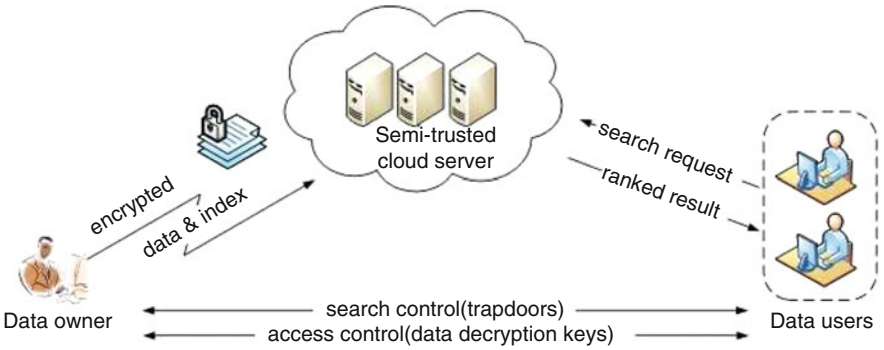

Fig. 1 Architecture of encrypted data search problem (From [10])

2 Overview of Search Over Encrypted Data

2.1 Problem Formulation

In this subsection, we will briefly introduce the general system model of the encrypted data search problem, its threat model and search privacy related requirements in the following.

System Model

The typical participants of a secure search system in the cloud involve the *cloud server*, the *data owner*, and the *data user*, as shown in Fig. 1. The data owner outsources the encrypted dataset and the corresponding secure indexes to the cloud server, where data can be encrypted using any secure encryption technique, such as Advanced Encryption Standard (AES), while the secure index is generated by some particular search-enabled encryption techniques. When a data user wants to query

the outsourced dataset hosted on the cloud server, he/she first either generates a trapdoor with the keyword of interest (applied to most PKC-based search schemes), or requests such trapdoor by sending a set of intended keywords to the data owner (in the case of SKC-based search schemes). In the latter case, upon receiving the trapdoor generation request, the data owner constructs the trapdoor, and return it to the user. Then the data user submits the trapdoor to the cloud server. The cloud server will execute the search program with the trapdoor as the input, the search results will be sent back to the user. Note that here we assume there is pre-existing security context between each user and the data owner thus authentication between user and data owner is already in place. The trapdoors can be requested and returned through a secure channel. The management of the decryption keys of the returned files is an orthogonal problem and has been studied separately [28, 48]. Search can be based on certain search criteria and the results be ranked based on certain ranking criteria so that the server returns all the matching documents or only the top-k most relevant ones to the user so as to realise effective and efficient data retrieval functionality, and mitigate the corresponding communication overhead, where k could be predefined by the user at the trapdoor submission time.

Threat Model

The typical threat model that most secure search schemes adopt [6, 10, 27, 39, 43] is to consider the cloud server to be "honest-but-curious", that is the cloud server "honestly" follows the designated protocol specification, but it is "curious" to infer and analyze data (including indexes) in its storage and message flows received during the protocol in order to learn additional information.

Search Privacy

In the literature, many privacy requirements are defined for PKC-based and SKC-based search schemes. We briefly introduce these *search privacy* requirements as follows.

1. **Keyword Privacy:** One of the major privacy concerns is how to protect the keywords of interest in a user's trapdoor against the cloud server. In other words, cloud server is not able to infer what the data user is searching. This fundamental privacy requirement should be satisfied for any valid encrypted data search scheme. Although trapdoor generation can be performed in a cryptographic way to protect the query keywords, the cloud server could identify the searched keywords by other side channel attacks, such as frequency analysis attack [39, 40, 43, 49]. Given the keyword-specific document frequency information (the number of documents containing the keyword) or the keyword frequency (the occurrence count of a keyword in a document) distribution information in a particular dataset, it is sufficient for an attacker to reverse-engineer the keyword in a trapdoor. Notice that this privacy requirement is referred to as *predicate*

privacy in the PKC-based search scenario and it cannot be protected inherently for any asymmetric secure search scheme [34].

2. **Trapdoor Unlinkability:** It is required that the trapdoor should be generated in a random manner. Otherwise, given any two trapdoors, the attacker can easily determine the relationship of them, such as whether they contain the same set of keywords. Therefore, sufficient nondeterminacy should be introduced into the trapdoor generation algorithm. It is worth noting that violation of this privacy requirement can further compromise the keyword privacy in that it allows the cloud server to accumulate frequencies of different search requests with respect to different keyword(s).
3. **Access Pattern:** It is defined to be the sequence of returned documents. Note that protecting access pattern by using private information retrieval technique [12,21] is extremely expensive since the algorithm has to "touch" the whole dataset outsourced on the cloud server which is inefficient in the large scale cloud system. Thus for efficiency concerns, most of the search over encrypted data schemes do not aim to protect it.

2.2 PKC-Based Search vs. SKC-Based Search

In PKC-based search schemes, different keys are used to generate index and trapdoor, such that a data user is usually free to produce a trapdoor by his/her keywords of interest without interacting with a data owner. Thus, this technique is much more suitable for some dynamic environment, e.g., when multiple data contributors and multiple data users exist in one search system. Otherwise, in SKC-based search, to search datasets from multiple data owners, a data user has to obtain these trapdoors from each individual data owner. This communication cost is definitely cumbersome to the user in such a multi-user and multi-data-contributor scenario. In addition, PKC-based search schemes can achieve more flexible and expressive queries compared with SKC-based schemes in general. For example, range query, subset query, etc., can be easily realised by PKC-based search schemes. In symmetric setting, how to generate trapdoors with similar functionalities is still a challenging problem.

It is worth noting that multi-keyword search can be achieved in both symmetric and asymmetric settings. By incorporating some ranking criteria, a data user is able to enjoy ranked search results by the relevance of documents to the query. Although conjunctive keyword search over encrypted data schemes in public-key setting also provide multi-keyword search function, it often lacks ranking functionality. Moreover, another unparalleled advantage of SKC-based search over PKC-based one is that the overall search process is much more efficient, since asymmetric search usually incurs a lot of time-consumed paring operations. Thus, there has been significant interest in developing efficient SKC-based encrypted data search mechanisms.

In what follows, we review some important related works built from either PKC or SKC technique.

PKC-Based Search

Inspired by identity-based encryption [8], Boneh et al. [7] propose the first PKC-based keyword search scheme with single keyword query, where anyone with public key can write to the data stored on server but only data users with private key can search. Following this work, a lot of PKC-based search schemes have been proposed to enrich the search functionalities. The scheme from [18] supports search queries with conjunctive keywords by explicitly indicating the number of encrypted keywords in an index, that is each keyword within a document is transformed to be a part of the index for this document. When doing query, the server should know which randomized keywords in the index need to be used for match evaluation. This information leakage may raise some privacy concerns. The authors in [20] also present a conjunctive keyword search over encrypted data scheme. They group the queried keywords together in an index to mitigate keyword privacy breach. But this is not flexible, since the data owner has to generate all the possible keyword combinations in one index. In addition, they extend the proposed secure keyword search to multi-user setting, where an encrypted index can be searched by various users holding different private keys. Predicate encryption (PE) [4,9,23,36] is another promising technique to fulfill the expressive search functionality over encrypted data. For example, the proposed scheme in [9] supports conjunctive, subset and range queries, and disjunctions, polynomial equations, and inner products could be realised in [23]. Li et al. [27] use the hierarchical predicate encryption technique to build an authorized keyword search scheme in the cloud. In their design, only authorized data users can be granted search capability, and unauthorized users are not allowed to search the dataset on the cloud server. Nevertheless, these PE-based secure search schemes are generally too computationally intensive to be implemented for practical use.

SKC-Based Search

Curtmola et al. design a symmetric secure search scheme supporting single keyword queries with security guarantees under rigorous definitions [15]. Owing to the adoption of inverted index [31] as the underlying index structure in their scheme, the search process can be extremely efficient. In [40, 43, 49], the order-preserving techniques are utilized to protect the rank order. By incorporating keyword frequency information and inverted index structure, they can achieve accurate and efficient search at the same time, but only single keyword query is supported. In addition, Kamara et al. [22] propose a dynamic version of [15] with the ability to add and delete files efficiently. In multi-user setting [6, 47], the authors separately present an encrypted data search schemes in the enterprise environment. Specifically, the data

user must be authorized before he/she can search the dataset, where authorization is enforced by a user list stored and managed by enterprise servers. Note that they differ from the PKC-based work [20] in terms of the allowed number of data contributors. Under the symmetric-key setting, merely one data contributor (enterprise) is allowed in their designs. In [17], the author formulates an IND-CKA security model for indexes, i.e., indistinguishability against chosen keyword attack, and a stronger model IND2-CKA. He also exploits pseudo-random functions and bloom filter to generate a secure index for each file, and its search time is proportional to the number of files in the dataset. The main problem of this scheme is that the final search results inevitably contain false positive due to bloom filter being the underlying index construction technique. Chang et al. [11] present a similar security model to IND2-CKA, and propose a secure search scheme with the index built from pseudo-random functions. Cao et al. [10] propose the first SKC-based encrypted data search scheme supporting multi-keyword ranked search where an index is generated using secure inner product technique. The ranking is realised by similarity measure of coordinate matching. Later, Sun et al. [39] present another secure multi-keyword search scheme in the cloud enabling more accurate search result ranking by using the state-of-the-art similarity measure, i.e., cosine measure in the vector space model, and design a search algorithm over the proposed tree-based index structure to fulfill more efficient search complexity in practice.

2.3 Exact Keyword Search vs. Fuzzy Keyword Search

Unlike the exact keyword search schemes above, it is common that keywords may be entered by a user which contain typos, but the search engine (e.g., Google search) is still capable of tolerating them and returning what the user intends. Thus, *fuzzy keyword search* technique is often used to rectify the mistakes. For the search algorithm to better understand the difference between a correct keyword and its typo, we need a similarity measurement to be supported in the underlying encrypted data search scheme, such that the matching files will be returned when the user's search request exactly matches the keyword in the index or the difference is within some predefined tolerance range.

Li et al. [26] propose a fuzzy keyword search over encrypted data scheme in the cloud. For each keyword, they first construct a wildcard keyword set containing all the variants of the keyword. Upon receipt of the trapdoor, they exploit edit distance to quantify keyword similarity. If the intended keyword is within the fuzzy keyword set, it will be considered a keyword match. By the similar techniques, Liu et al. [29] present another fuzzy keyword search scheme with a size-reduced index. In [13], the authors use a B-tree [14] based index structure to construct a fuzzy keyword search scheme, where although they claim the support of multiple keyword search capability, they only group several keywords together to form a phrase. This is analog to some conjunctive keyword search schemes [20] in public-key setting,

which is apparently not flexible in the sense that a data user is not able to query any combination of keywords of his/her choice if this keyword combination is not considered by the data owner at the index generation phase.

2.4 Secure Index-Based Search

Since Song et al.'s seminal work [38], searchable encryption has drawn a lot of attention. Their work enables in-line text search within an encrypted document. Specifically, their scheme encrypts each document word by word and performs full-domain search such that it takes linear operations to cover all the documents. Later, to improve the search efficiency, many secure search schemes have been proposed, where queries can be executed over encrypted indexes (rather than encrypted data themselves) by users who possess proper "trapdoors". This applies to all the encrypted data search schemes mentioned above. To design a keyword search over encrypted data scheme, a series of important factors should be considered as follows.

Index Structure

In general, there are three kinds of index structures often used to construct encrypted data search schemes:

- **Index organized by keywords:** Such index data structure is usually called *inverted index*, or *inverted file* [31]. In this data structure, each keyword is followed by a file list which consists of all the files in the dataset containing this keyword. The advantage of this index structure is to allow significantly efficient text search instead of the full domain text search. But when a file is added to the dataset, it needs increased processing since all the indexes containing the keywords in this file have to be updated. This kind of index structure is widely used in encrypted data search schemes [15,22,40,43,49] due to its extremely fast search process. Note that search schemes with this keyword-based index structure can only realise single keyword query.
- **Index constructed per document:** Another popular index structure adopted by many secure search schemes [10, 11, 17, 20] is to construct an index for each file in the dataset such that one file update usually affects only one corresponding index. The index structure is specific to each document and is generated by all the keywords contained in the target document.
- **Tree-based index structure:** Index structure can also be constructed based on some well-developed tree structures, such as *B*-tree [14], MDB-tree [33], etc. A few existing works [30,39] exploit tree-based structures to design efficient secure search schemes in different scenarios.

Secure Search Algorithm

According to different data structures, search over encrypted data schemes may use different secure search algorithm to do the match. The inverted index structure allows fast direct intended file retrieval, so the search complexity is constant there. For example, the indexed keywords can be hashed and then store the associated file list at a table with its address being the hash value . When a user wants to search a keyword of interest, he/she first hashes it and submits the hash value to the server. Therefore, the server is able to find out the intended files efficiently.

For schemes with index built from each document, the most efficient search algorithm merely enables linear search, i.e., the time for search is linear to the number of documents in the dataset, since the returned search results could not be determined until the search process goes through all the indexes within the document set. This is not desirable when a huge amount of data are present on the server.

By utilizing tree-based structures to construct indexes for encrypted data search schemes, the corresponding secure search algorithm could be devised to achieve more efficient search than the linear search schemes. At the meantime, the same expressive queries as the schemes with index built per document could be realised under this index structure, such as range queries in database scenario [30] and multi-keyword text search with similarity-based ranking [39].

Similarity-Based Ranking

To enhance user searching experience and meet more effective data retrieval need, two fundamental aspects have to be considered when designing a practical encrypted data search scheme. On one hand, most of today's search engines on the Internet (e.g., Google search) allow users to query multiple keywords in one search request instead of only one as the indicator of their search interest. Compared with single keyword query, the main advantage of this multi-keyword search is that it can yield more relevant search results efficiently. On the other hand, ranked search functionality is preferable in the "pay-as-you-go" cloud paradigm. The reason is that cloud server could conduct relevance ranking operation for data user and return the most relevant set of files, rather than directly sending back the undifferentiated search results to data user. As such, the network traffic between cloud server and data user could be dramatically reduced.

By securely incorporating advanced similarity measures into the design of encrypted data search schemes, ranking functionality could be realised during search process with a multi-keyword trapdoor. These adopted similarity measures are borrowed from plaintext information retrieval community, such as coordinate matching, cosine measure in the vector space model [45]. As a result, the constructed encrypted data search schemes enjoys the same flexibility and search result accuracy as the existing multi-keyword search over plaintext.

3 Privacy-Preserving Multi-keyword Ranked Search

Cao et al. [10], for the first time, explore the problem of multi-keyword ranked search over encrypted cloud data (MRSE), and establish a set of strict privacy requirements for such a secure cloud data utilization system. They propose two MRSE schemes based on the similarity measure of coordinate matching while meeting different privacy requirements in two different threat models. One is *known ciphertext model*, where the cloud server is supposed to only know encrypted dataset and searchable index, both of which are outsourced from the data owner. The other is *known background model*, in which the cloud server could possess more knowledge than what can be accessed in the known ciphertext model, such as document frequency information. At the meantime, they execute thorough security analysis and experiment evaluation on the real world dataset to demonstrate the privacy and efficiency guarantees of their proposed schemes. In the remaining of this section, we will discuss this work.

3.1 Technical Overview for MRSE

Coordinate Matching

To support multi-keyword ranked search, the similarity measure, *coordinate matching* [45], is incorporated into the MRSE schemes. This similarity measure counts the number of query keywords appearing in the documents to quantify the relevance of that document to the query. The more query keywords that appear in a document, the more relevant the document to the query. This similarity measure is thought of as a hybrid intermediate between conjunctive and disjunctive search. Any document with all or partial keywords matching is considered a part of the search results. To formalize such similarity measure in practice, they use inner products of the query vector and a set of document index vectors to reflect the predilection of the data user for documents. For example, assume that a dictionary is defined as $\{search, cloud, privacy, network, security\}$. There are two documents A, B in the dataset. Therefore, set the index vector as a binary vector $D_A = (1,0,0,1,1)$ for document A if it only contains keywords $\{search, network, security\}$, where 1 is used to indicate the existence of some keyword in the document and 0 otherwise. If the keywords $\{search, cloud, security\}$ appears in the document B, the binary index vector D_B is defined to be $(1,1,0,0,1)$. Suppose that the data user has a query with the intended keywords $\{seach, cloud, privacy\}$. Thus the binary query vector Q is represented as $(1,1,1,0,0)$. We can calculate the inner products of the query vector Q and the index vectors D_A, D_B as the similarity scores of documents A and B:

$$SimilarityScore_A = Q \cdot D_A = (1,1,1,0,0) \cdot (1,0,0,1,1) = 1,$$

and

$$SimilarityScore_B = Q \cdot D_B = (1,1,1,0,0) \cdot (1,1,0,0,1) = 2.$$

Therefore, we can deduce that the data user would prefer document B to document A since the similarity score of B is greater than that of A. Also, it yields a ranking $B > A$.

By using the coordinate matching similarity measure, effective multi-keyword ranked search functionality could be realised. Nevertheless, such measure is originally designed for plaintext information retrieval purpose. How to apply it to the encrypted data search without breaching search privacy is a very challenging problem.

Search with Secure Inner Product Evaluation

To use the above mentioned similarity measure in a privacy-preserving way, index vector D_d for each document d, query vector Q and their inner product $D_d \cdot Q$ should not be exposed to the cloud server. In MRSE, the authors propose a secure inner product scheme which is adapted from a secure k-nearest neighbor (kNN) technique [46] to hide these sensitive information.

In database scenario, secure kNN technique can be exploited to select k nearest database records to the query by comparing the Euclidean distance between them. Specifically, each record in the database and the query can be represented by an n-dimensional vectors p_i and q respectively. The secret key consists of one $(n+1)$-dimensional vector S and two $(n+1) \times (n+1)$ invertible matrices M_1 and M_2. Then after vector extension, a new p_i is set as $(p_i, -0.5||p_i^2||)$ and a new query vector q is (rq, r), where $r > 0$ is a random number. As per the splitting indicator S, p_i is split into two vectors as $\{p_i', p_i''\}$ and q is also split into two vectors $\{q', q''\}$ such that p_i and q can be recovered given S, $\{p_i', p_i''\}$ and $\{q', q''\}$. Eventually, the vector pairs $\{p_i', p_i''\}$ and $\{q', q''\}$ are encrypted as $\{M_1^T p_i', M_2^T p_i''\}$ and $\{M_1^{-1} q', M_2^{-1} q''\}$ respectively. At the database search phase, the product of encrypted record vector pair and encrypted query vector pair, i.e., $-0.5r(||p_i||^2 - 2p_i \cdot q)$, is serving as the indicator of Euclidean distance $(||p_i||^2 - 2p_i \cdot q + ||q||^2)$ to select k nearest neighbors. Without prior knowledge of secret key, neither record vector nor query vector, after such a series of processes, can be recovered by analyzing their corresponding ciphertext.

Cao et al. modify this secure kNN technique to measure the inner product similarity instead of the Euclidean distance. In particular, trapdoor vector Q is extended to be (rQ, r, t), where r, t are two random numbers and $r > 0$, such that it is difficult for the cloud server to infer the relationship among the received trapdoors. To obfuscate the document frequency and diminish the chances for re-identifying the keywords, the final similarity scores should be further randomized. Thus some randomness ε_d is introduced into the index vector D_d, and D_d is extended into

$(D_d, \varepsilon_d, 1)$. The encrypted index vector pair $I_d = \{M_1^T D'_d, M_2^T D_d''\}$ and trapdoor vector pair $T = \{M_1^{-1} Q', M_2^{-1} Q''\}$ are generated after applying the vector splitting and matrix multiplication. The final similarity score for document d to the query vector would be:

$$\begin{aligned} I_d \cdot T &= \{M_1^T D'_d, M_2^T D''_d\} \cdot \{M_1^{-1} Q', M_2^{-1} Q''\} \\ &= D'_d \cdot Q' + D''_d \cdot Q'' \\ &= (D_d, \varepsilon_d, 1) \cdot (rQ, r, t) \\ &= r(D_d \cdot Q + \varepsilon_d) + t. \end{aligned}$$

By using this equation, the ranked search result can be produced.

This vector encryption method has been proved to be secure in the known ciphertext model [46]. As long as the secret key is kept confidential, the underlying plaintext information in the index vector and trapdoor vector cannot be revealed.

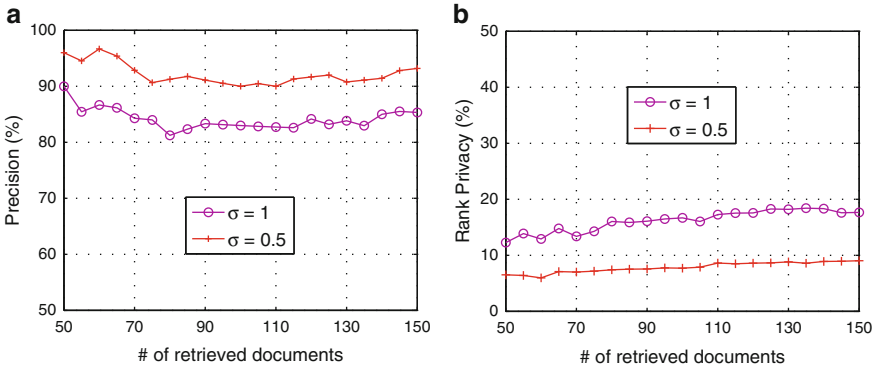

Fig. 2 Tradeoff between (**a**) precision, and (**b**) rank privacy by selecting different standard deviation σ (From [10])

Note that, let ε_d follow a Normal distribution $N(\mu, \sigma^2)$, where the standard deviation σ functions as a flexible trade-off parameter between search accuracy and security. To protect *keyword privacy*, a large σ is selected to introduce more obfuscation into the final similarity score, from which it is difficult for the cloud server to gain statistical information about the original similarity score, but the search result could be less accurate. Thus from the viewpoint of the effective search, small σ is preferable. This is shown in Fig. 2.[1] Due to the splitting process and the random

[1]Precision is defined to be the fraction of returned top-k documents that are included in the real top-k list, while rank privacy measures the rank order variation between the returned top-k documents and real top-k documents.

numbers r,t, the trapdoor generation algorithm can output two different trapdoors even for the same search request to guarantee *trapdoor unlinkability*.

To further protect search privacy in the known background model, an enhanced MRSE scheme is proposed. The main modification is to insert more dummy keywords $\sum \varepsilon_d$ instead of only one fixed ε_d into the index vector for each document. The level of search accuracy remains the same with the previous basic MRSE scheme if let $\sum \varepsilon_d$ follow a Normal distribution as well.

Cao et al. for the first time, define and solve the problem of multi-keyword ranked search over encrypted cloud data by combining the efficient similarity measure "coordinate matching" with the adapted secure inner product technique. The proposed schemes can meet various stringent privacy requirements while retaining effective search functionalities.

4 Improvement on Search Accuracy and Efficiency

4.1 Background

Although MRSE can achieve multi-keyword ranked search, there exists a gap between MRSE and the state-of-the-art plaintext information retrieval techniques in terms of search accuracy and search efficiency. On one hand, the similarity measure "coordinate matching" in MRSE has some drawbacks when used to evaluate the document ranking order. First, it takes no account of term[2] frequency such that any keyword appearing in a document will present in the index vector as binary value 1 for that document, irrespective of the number of its appearance. Obviously, it fails to reflect the importance of a frequently appeared keyword to the document. Second, it takes no account of term scarcity. Usually a keyword appearing in only one document is more important than a keyword appearing in several ones. In addition, long documents with many terms will be favored by the ranking process because they are likely to contain more terms than short documents. Hence, due to these limitations, the heuristic ranking function, "coordinate matching", is not able to produce more accurate search results. More advanced similarity measure should be adopted from plaintext information retrieval community, such as *cosine measure* in the *vector space model* [45]. On the other hand, the search complexity of MRSE is linear to the number of documents in the dataset, which becomes undesirable and inefficient when a huge amount of documents are present, while many efficient index structures exist in the plaintext information retrieval techniques, e.g., B-tree [14], inverted index [31], etc.

Sun et al. [39] present a privacy-preserving multi-keyword text search (MTS) scheme in the cloud supporting similarity-based ranking to address the challenge of

[2]We do not differentiate *term* and *keyword* hereafter.

constructing more accurate, practically efficient and flexible encrypted data search functionalities. Specifically, the index vector for each document is generated based on the cosine measure in the vector space model to support multi-keyword query and search result ranking functionality, and utilize the "term frequency (TF) × inverse document frequency (IDF)" weight to achieve high search result accuracy. By incorporating the state-of-the-art information retrieval technique, the proposed MTS schemes enjoy the same flexibility and search result accuracy as the existing state-of-the-art multi-keyword ranked search over plaintext. In order to improve the search efficiency, they propose a tree-based index structure, where each value in a node is a vector of term frequency related information. Furthermore, an efficient search algorithm is presented to realise more efficient search functionality compared with [10]. To satisfy various search privacy requirements, two secure index schemes for multi-keyword text search with similarity-based ranking are devised. The basic scheme (BMTS) is secure under the known ciphertext model, and the other enhanced secure index scheme (EMTS) is constructed against sensitive frequency information leakage to meet more stringent privacy requirements under the stronger threat model, i.e., known background model.

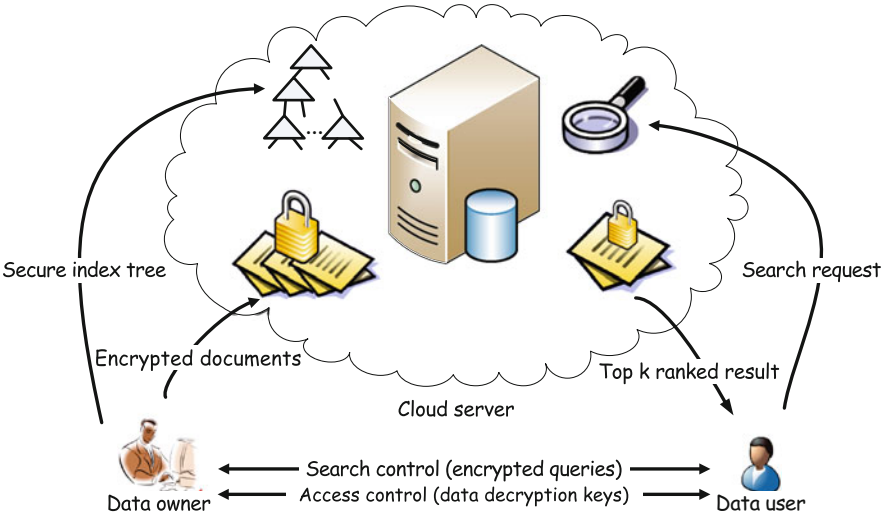

Fig. 3 Framework of MTS (From [39])

4.2 Technical Overview of MTS

The system framework in [39] is analog to [10] as shown in Fig. 3, wherein three participants, i.e., the data owner, the data user and the cloud server, are defined. Note that the index vectors are organized as a secure index tree instead of each individual vector before outsourced to the cloud server. Assume the cloud server still acts in an

"honest-but-curious" manner. Since the term frequency information is incorporated into the ranking function, in the known background model the attacker may extract such statistical information from a known comparable dataset of the similar nature to the target 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 particular keywords in the query [40, 43, 49].

Vector Space Model

Vector space model is one of the most popular similarity measures in the plain-text information retrieval community, supporting both conjunctive and disjunctive search. The ranking order for a particular document set is determined by comparing the deviation of angles, i.e., cosine values, between each document vector and the query vector. The cosine measure allows accurate ranking due to the "TF×IDF rule", where TF denotes the occurrence count of a term within a document,[3] and IDF is obtained by dividing the total number of documents in the collection by the number of documents containing the term.[4] Thus, unlike the coordinate matching, each dimension of an index vector in MTS is a TF weight $w_{d,t}$, and a query vector is comprised of IDF weights $w_{q,t}$, where d, t denote a specific document in the dataset and a term in the dictionary respectively. The ranked search functionality can be achieved by the following similarity function:

$$Cos(D_d, Q) = \frac{1}{W_d W_q} \sum_{t \in Q \cap D_d} w_{d,t} \cdot w_{q,t},$$

where $W_d = \sqrt{\sum_{t \in Q \cap D_d} w_{d,t}^2}$, $W_q = \sqrt{\sum_{t \in Q \cap D_d} w_{q,t}^2}$. Thus, the index vector D_d and query vector Q are both unit vectors.

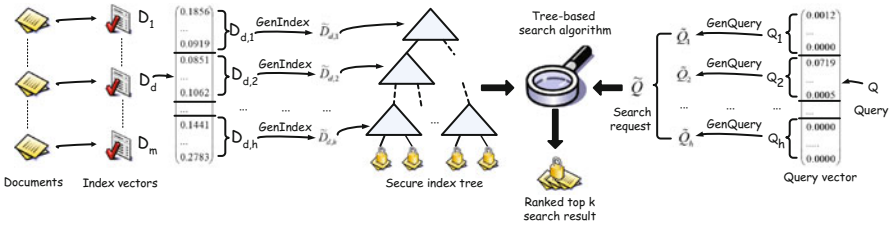

Fig. 4 Overview of secure index scheme (From [39])

[3] It is used to measure how important a specific term is to a particular document.

[4] It implies that this frequency of a term tends to be inversely proportional to its ranking.

Secure Index Scheme

To construct the index tree structure, the original long document index vector D_d has to be divided into multiple sub-vectors such that each sub-vector $D_{d,i}$ represents a subset of keywords, and becomes a part of the i-th level of the index tree, as shown in Fig. 4. Similarly, let Q_i be the query sub-vector at the i-th 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. The similar secure inner product scheme [10] is adopted here but applied to each level of the index tree. In addition, they do not use the dimension extension technique for BMTS in the known ciphertext model. The similarity score at the i-th level is computed as follows:

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

where $\widetilde{D_{d,i}}$ and $\widetilde{Q_i}$ represent the encrypted forms of index vector and query vector at the i-th level respectively. Hence, the final similarity score for document d is $\sum_{i=1}^{h} D_{d,i} \cdot Q_i = D_d \cdot Q$ by assuming that the index tree has h levels in total.

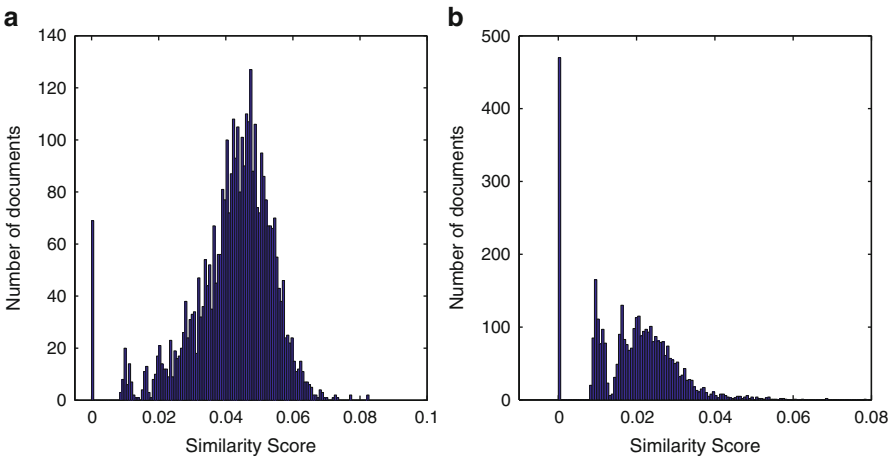

Fig. 5 Distribution of similarity score when a single keyword in a query vector with BMTS. (**a**) For keyword "network". (**b**) For keyword "search" (From [39])

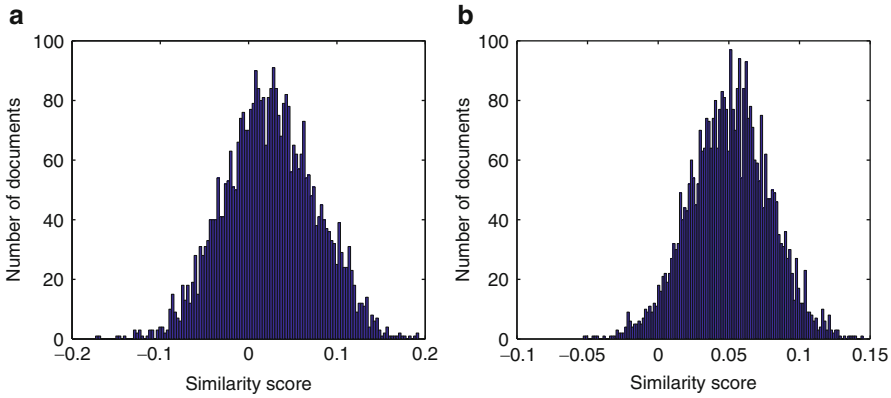

Fig. 6 Obfuscation to distribution of similarity score for keyword "network" with different standard deviations in EMTS. (**a**) $\sigma = 0.05$. (**b**) $\sigma = 0.03$ (From [39])

In BMTS, index and query confidentiality can be well protected by the secure inner product technique. Due to the non-deterministic property of the encryption method, the trapdoor unlinkability can be preserved similar to [10]. As assumed in the defined known background model, the cloud server may have the knowledge of the TF distributions, or normalized ones of some sensitive keywords from a known comparable dataset. It is worth noting that these distributions are keyword specific, as shown in Fig. 5.[5] Therefore, to further prevent this sensitive information from being disclosed to the server, the authors insert phantom terms into the query vector in EMTS so as to obfuscate the final similarity scores while maintaining effective search functionalities, as shown in Fig. 6. The larger σ is selected, the better the TF distribution can be protected. This technique can achieve the same privacy preserving functionality as MRSE, and the selection of σ reflects the user's preference for privacy preservation or search accuracy. On the other hand, this query-side randomization technique significantly differs from [10] in the sense that randomization in [10] is applied to the index vector and is not possible to be calibrated by users as an effective privacy-preserving parameter.

Efficient Tree-Based Search Algorithm

In database community, query process could complete in logarithmic time by using B-tree, B^+-tree, etc. These tree-based structures are not only used in the plaintext database search, but also can be used in the encrypted database scenario [30] to realise efficient range query. Nevertheless, they are not applicable to text search. The similarity score is a dynamic value depending on the query and has to be evaluated

[5]The background dataset is collected from the recent 10 years' IEEE INFOCOM publications.

in the runtime, which makes the fixed tree structure, such as B-tree or B^+-tree, not suitable here. Inverted index [31] is the most efficient and well-developed index structure which is widely used in the plaintext information retrieval community. In the literature, however, a few works [15, 40, 43, 49] employ this technique to design efficient search algorithm but only for single keyword query. Sun et al. propose a tree-based search algorithm, which is adapted from multi-dimensional B-tree (MDB-tree) [33] based multi-dimensional algorithm (MD-algorithm) [32], to enable efficient multi-keyword ranked search.

The MD-algorithm is used to find the k-best matches in a database that is structured as an MDB-tree, as shown in Fig. 7. Each attribute domain in the database constitutes one level of the MDB-tree 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. 7, $\hat{P}_i = P_i = 1.0$.

In a depth-first manner, MD-algorithm starts from the root node with a recursive procedure upon this tree. Specifically, search process selects the unused maximum attribute value when it enters a node, and based on $\hat{P}_i$'s below this level, predicts the maximum possible final score to be obtained. The criteria for node selection is that if this predicted final score is less than or equal to the minimum score of the top-k objects which have been selected, search process returns to the parent

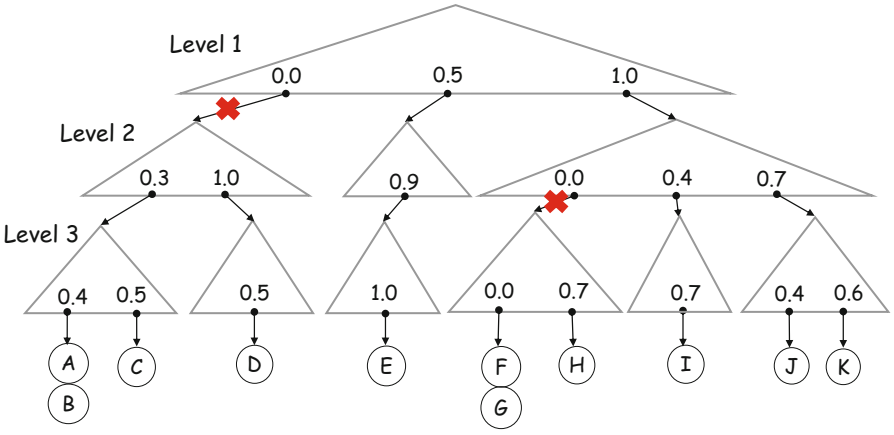

Fig. 7 Illustration of MD-algorithm on MDB-tree (From [39])

node, otherwise, it goes down to the child node at the next level. This procedure is executed recursively until the objects with top-k scores are selected.

The search can be done very efficiently due to the relatively accurate final score prediction, and thus only part of the objects in the tree are accessed. Figure 7 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. More details of the MD-algorithm and MDB-tree can be found in [32].

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, the index tree structure has to be built on vectors. Another remarkable difference between the proposed search algorithm and MD-algorithm is that it is not possible to set $\hat{P}_i$ to P_i as running the MD-algorithm in database scenario, since P_i varies for queries and has to be securely evaluated in the runtime.

Search Efficiency Improvements

During the evaluation of the MD-algorithm on the proposed secure index tree, three important efficiency-improving factors are identified by the authors. Next, we will briefly elaborate on those observations.

1. **Impact of Prediction Threshold Value:** By observation, they found that the smaller the predication threshold value, the faster the search algorithm is terminated, which means the search process can be terminated earlier without going into unnecessary nodes. As such, at each level, $\hat{P}_i$ should decreasingly approach P_i as close as possible.

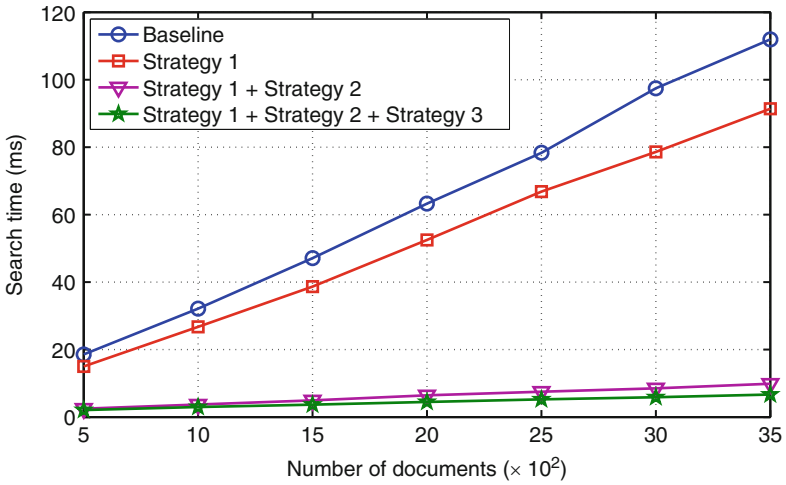

Fig. 8 Comparison of search efficiency with different efficiency-improving strategies (From [39])

2. **Impact of Intended Keyword Position:** Another observed efficiency-improving factor is that the search efficiency is significantly dependent on the position of the intended keywords on the index tree. Indeed, people usually complete a search with a query only consisting of a few keywords [1], which is different from using the MD-algorithm in database scenario. Typically, to find out the object of interest, all the attributes are utilized to query the database. It is apparently inefficient since the search process needs to go to the bottom level of the index tree where the intended attribute resides.
3. **Impact of Index Vector Clustering:** The last search efficiency related observation is that "similar" vector index could be clustered together to reduce the number of accessed nodes in the index tree at the expense of lower search precision.

Based the these key observations, the authors propose the corresponding effective strategies to improve the practical search efficiency with vector indexes while not introducing new privacy vulnerabilities. Compared with the original MD-algorithm, the experimental result[6] shows the much more improved search efficiency in Fig. 8.[7] Furthermore, Fig. 9a shows the search time for BMTS and EMTS with the proposed efficiency-improving strategies, compared with [10] and baseline search with respect to the size of document set. Due to the proposed search algorithm and tree-based index structure, the baseline search is far efficient than [10]. Note that the time cost of BMTS and EMTS is more efficient than [10] and the baseline search. Besides, the two proposed schemes enjoy almost the same and nearly constant search time. Figure 9b shows that the proposed secure search schemes are still extremely efficient in the case of more documents are required to be returned.

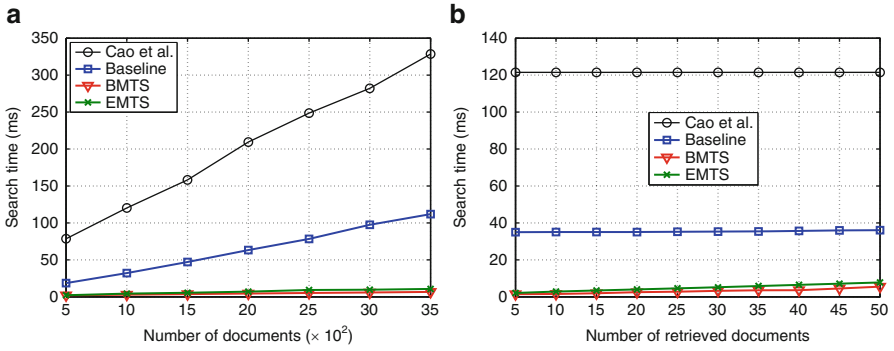

Fig. 9 Comparison of search efficiency with the same 10 keywords of interest. (**a**) For the different size of document set. (**b**) For the different number of retrieved documents (From [39])

[6]All the experimental results in [39] are obtained from implementation of the proposed secure search system using JAVA on a Linux Server with Intel Core i3 Processor 3.3 GHz.

[7]The baseline search is with respect to the original MD-algorithm. The strategies 1 is proposed from the observation 1. Likewise, the strategy 2 is from the observation 2 and the strategy 3 from the observation 3.

5 Conclusion

With the advent of cloud computing, more and more sensitive data are outsourced to the cloud server to reduce the management cost and enjoy the ubiquitous access. However, this novel computing paradigm introduces serious privacy challenges in that users' data are no longer locally possessed but stored on the remote server which belongs to a different trust domain compared with the data users'. In this chapter, we focus on the privacy concerns in the secure search function performed over encrypted cloud data. We first provide a brief introduction to the background knowledge of encrypted data search techniques that have been proposed in the literature and dedicated to address the secure search problem in the computation outsourcing model. Then we elaborate on a state-of-the-art secure search scheme in the text search scenario, and show that they can achieve flexible/expressive search functionalities, i.e., multi-keyword ranked search. In addition, the same search accuracy as the plaintext information retrieval can be realised using the state-of-the-art similarity measure while search privacy is well protected. Finally, with the proposed search algorithm, the discussed secure search system is efficient enough to be deployed in practice.

While continued research is necessary to further enrich the search functionality and improve the efficiency and scalability of search schemes, another very interesting direction is on virtualization security that tries to secure the execution environment (i.e., virtual machines) in the cloud server. This will require a slight change in the security model – instead of an honest-but-curious server model which does not trust the server, we may choose to place minimum trust on the server, for example, trust the bare hardware on the server, and design the secure operating system to protect the virtual machine against the software-based attacks, be it from other virtual machines running on the same physical machine or the hosting machine's operating system. We argue that the data should be stored in the cloud in the encrypted form. However, after they are loaded to the users' secure execution environment, they can be in the plaintext form in order to enable effective computation, such as search. Research along this line includes [5, 24, 41, 50] and it aims to provide a more general solution to the secure computation on the untrusted cloud server problem. We believe both research directions are interesting and call for more effort from the research community.

Acknowledgments 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 grant CNS-1217889.

References

1. Keyword and search engines statistics. http://www.keyworddiscovery.com/keyword-stats.html?date=2013-01-01 (2013)
2. Atallah, M.J., Frikken, K.B.: Securely outsourcing linear algebra computations. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 48–59. ACM (2010)
3. Atallah, M.J., Li, J.: Secure outsourcing of sequence comparisons. International Journal of Information Security **4**(4), 277–287 (2005)
4. Attrapadung, N., Libert, B.: Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In: Public Key Cryptography–PKC 2010, pp. 384–402. Springer (2010)
5. Azab, A.M., Ning, P., Zhang, X.: Sice: a hardware-level strongly isolated computing environment for x86 multi-core platforms. In: Proceedings of the 18th ACM conference on Computer and communications security, pp. 375–388. ACM (2011)
6. Bao, F., Deng, R.H., Ding, X., Yang, Y.: Private query on encrypted data in multi-user settings. In: Information Security Practice and Experience, pp. 71–85. Springer (2008)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology-Eurocrypt 2004, pp. 506–522. Springer (2004)
8. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Advances in Cryptology – CRYPTO 2001, pp. 213–229. Springer (2001)
9. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Proceedings of the 4th conference on Theory of cryptography, pp. 535–554. Springer-Verlag (2007)
10. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Proceedings of IEEE INFOCOM, pp. 829–837 (2011)
11. Chang, Y.C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Applied Cryptography and Network Security, pp. 442–455. Springer (2005)
12. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. Journal of the ACM **45**(6), 965–981 (1998)
13. Chuah, M., Hu, W.: Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In: Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on, pp. 273–281. IEEE (2011)
14. Comer, D.: Ubiquitous b-tree. ACM computing surveys **11**(2), 121–137 (1979)
15. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on Computer and communications security, pp. 79–88. ACM (2006)
16. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
17. Goh, E.J.: Secure indexes. Cryptology ePrint Archive. http://eprint.iacr.org/2003/216 (2003)
18. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: ACNS 04: 2nd International Conference on Applied Cryptography and Network Security, pp. 31–45. Springer-Verlag (2004)
19. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Theory of Cryptography, pp. 264–282. Springer (2005)
20. Hwang, Y.H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Pairing-Based Cryptography–Pairing 2007, pp. 2–22. Springer (2007)
21. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography from anonymity. In: the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 239–248. IEEE (2006)
22. Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM conference on Computer and communications security, pp. 965–976. ACM (2012)
23. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Advances in Cryptology–EUROCRYPT 2008, pp. 146–162. Springer (2008)

24. Keller, E., Szefer, J., Rexford, J., Lee, R.B.: Nohype: virtualized cloud infrastructure without the virtualization. In: ACM SIGARCH Computer Architecture News, vol. 38, pp. 350–361. ACM (2010)
25. Krebs, B.: Payment processor breach may be largest ever. http://voices.washingtonpost.com/securityfix/2009/01/payment_processor_breach_may_b.html (2009)
26. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: INFOCOM, 2010 Proceedings IEEE, pp. 1–5. IEEE (2010)
27. Li, M., Yu, S., Cao, N., Lou, W.: Authorized private keyword search over encrypted data in cloud computing. In: Distributed Computing Systems (ICDCS), 2011 31st International Conference on, pp. 383–392. IEEE (2011)
28. Li, M., Yu, S., Zheng, Y., Ren, K., Lou, W.: Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Transactions on Parallel and Distributed Systems **24**(1), 131–143 (2013)
29. Liu, C., Zhu, L., Li, L., Tan, Y.: Fuzzy keyword search on encrypted cloud storage data with small index. In: Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on, pp. 269–273. IEEE (2011)
30. Lu, Y.: Privacy-preserving logarithmic-time search on encrypted data in cloud. In: 19th Annual Network and Distributed System Security Symposium (NDSS Symposium'12) (2012)
31. NIST: NIST's dictionary of algorithms and data structures: inverted index. http://xlinux.nist.gov/dads/HTML/invertedIndex.html
32. Ondreička, M., Pokornỳ, J.: Extending fagin's algorithm for more users based on multidimensional b-tree. In: Advances in Databases and Information Systems, pp. 199–214. Springer (2008)
33. Scheuermann, P., Ouksel, M.: Multidimensional b-trees for associative searching in database systems. Information systems **7**(2), 123–137 (1982)
34. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Theory of Cryptography, pp. 457–473. Springer (2009)
35. Sheridan, J., Cooper, C.: Defending the cloud. http://www.reactionpenetrationtesting.co.uk/Defending%20the%20Cloud%20v1.0.pdf (2012)
36. Shi, E., Bethencourt, J., Chan, H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 350–364 (2007)
37. Slocum, Z.: Your google docs: Soon in search results? http://news.cnet.com/8301-17939_109-1035713%207-2.html (2009)
38. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
39. Sun, W., Wang, B., Cao, N., Li, M., Lou, W., Hou, Y.T., Li, H.: Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. In: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, pp. 71–82. ACM (2013)
40. Swaminathan, A., Mao, Y., Su, G.M., Gou, H., Varna, A.L., He, S., Wu, M., Oard, D.W.: Confidentiality-preserving rank-ordered search. In: Proceedings of the 2007 ACM Workshop on Storage Security and Survivability, pp. 7–12 (2007)
41. Szefer, J., Keller, E., Lee, R.B., Rexford, J.: Eliminating the hypervisor attack surface for a more secure cloud. In: Proceedings of the 18th ACM conference on Computer and communications security, pp. 401–412. ACM (2011)
42. Van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Secure Data Management, pp. 87–100. Springer (2010)
43. Wang, C., Cao, N., Ren, K., Lou, W.: Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Transactions on Parallel and Distributed Systems **23**(8), 1467–1479 (2012)
44. Wang, C., Ren, K., Wang, J.: Secure and practical outsourcing of linear programming in cloud computing. In: INFOCOM, 2011 Proceedings IEEE, pp. 820–828. IEEE (2011)

45. Witten, I.H., Moffat, A., Bell, T.C.: Managing gigabytes: Compressing and indexing documents and images. Morgan Kaufmann Publishing, San Francisco, May 1999
46. Wong, W.K., Cheung, D.W.l., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, pp. 139–152. ACM (2009)
47. Yang, Y., Lu, H., Weng, J.: Multi-user private keyword search for cloud computing. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp. 264–271. IEEE (2011)
48. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of IEEE INFOCOM, pp. 1–9 (2010)
49. Zerr, S., Olmedilla, D., Nejdl, W., Siberski, W.: Zerber+ r: Top-k retrieval from a confidential index. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 439–449. ACM (2009)
50. Zhang, N., Li, M., Lou, W., Hou, Y.T.: Mushi: Toward multiple level security cloud with strong hardware level isolation. In: MILITARY COMMUNICATIONS CONFERENCE, 2012-MILCOM 2012, pp. 1–6. IEEE (2012)