Computer Communications 126 (2018) 50–63

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

PECS: Privacy enhanced conjunctive search over encrypted data in the cloud supporting parallel search

MB Smithamol*, Rajeswari Sridhar

Department of Computer Science and Engineering, Anna University, Chennai 600 025, India

ARTICLE INFO

Keywords:
Cloud computing
Partitioned index
Conjunctive search
Membership balanced binary tree (MBB)
Data outsourcing

ABSTRACT

The cost-effective storage and computing services offered by the cloud model have accelerated the rate of data outsourcing in domains like health industry and scientific research. Although cloud services provide multiple benefits, outsourcing sensitive information demands the assurance of security and privacy given that data and its access are not in control of the data owner. Searchable encryption is an active research area in the cloud model to improve the usage of outsourced encrypted data. In this paper, we propose a novel privacy enhanced conjunctive search (PECS) over encrypted data in the cloud supporting parallel search and dynamic updating to achieve search efficiency. To achieve this, first, we present a new tree-based partitioned index structure (TPIS) enabling parallel search to utilize the cloud resources efficiently. Parallel search is needed to retrieve data from the extensive collection of files outsourced. Second, we present two search schemes to provide query privacy and keyword privacy in the threat model. The proposed search method PECS satisfies the security and privacy goals of multi-keyword search over encrypted data. The results of our experiments show better search efficiency, indicating that the proposed conjunctive search scheme is suitable for practical use.

1. Introduction

Cloud computing has evolved as a successful paradigm with hypothetically unlimited storage and computation capacity, and advantages of the cloud model are prompting many enterprise tenants to find more innovative and cost-effective solutions to the problem of outsourcing data [1–3]. The database outsourcing to the cloud bring issues regarding possible data loss, dishonest utilization of sensitive information as the owners do not possess any direct control over the data. While more and more data has been outsourced to the cloud, security and privacy assurance is important to ensure that data is well protected. Despite the advantages of the cloud model, outsourcing brings new security challenges and risks [4–6].

The files are usually encrypted using secure cryptographic algorithms, and the encrypted files are outsourced to protect it from the cloud data service provider (CDSP). However, this will induce substantial computational cost regarding data usage. The major technique widely used for document retrieval is keyword search which cannot be directly employed for encrypted files [7]. To achieve secure outsourcing and retrieval, researchers have proposed solutions based on full homomorphic encryption and oblivious RAM [8]. However, these methods are not widely employed in the real environment due to their heavy computation and communication overhead. A more practical approach towards searching over encrypted data is searchable encryption [9].

Searchable encryption schemes enabled the data owner to perform keyword searches over the outsourced data. As an encryption system it should meet cryptographic properties like access pattern privacy, keyword privacy, consistency and resisting keyword guessing attack [10]. Many search frameworks proposed in the literature use either symmetric encryption or asymmetric encryption with different search options such as ranked keyword search, multi-user data sharing, multi keyword search and boolean search. Recently, security issues concerning with outsourcing of sensitive information such as health records and financial data sought more attention [11,12]. Search over such sensitive outsourced data demands the retrieval of all matching documents given a set of query words. For example, getting the health records of a person with keywords as lab records, the server is supposed to return all records containing lab results. Here, simple conjunctive query search is needed. The existing conjunctive search schemes use public key cryptosystem which are not scalable due to the computationally heavy bilinear pairing operation.

In this paper, we propose an efficient Privacy Enhanced Conjunctive Search (PECS) system using the novel indexing scheme, Tree-based Partition Index Structure (TPIS), over the encrypted data in the cloud. An important contribution in any searchable encryption scheme is the

* Corresponding author.
E-mail addresses: smithamolm@acm.org (M. Smithamol), rajisridhar@gmail.com (R. Sridhar).

https://doi.org/10.1016/j.comcom.2018.05.008
Received 13 October 2017; Received in revised form 23 April 2018; Accepted 17 May 2018
Available online 24 May 2018

construction of an efficient index structure. The proposed indexing scheme TPIS achieves sublinear search efficiency with minimum computation overhead. Specifically, membership balanced binary (MBB) tree is used as the base structure for the TPIS indexing. For every document in the collection, a set of relevant keywords is defined, and the documents in the collection are randomly mapped into different partitions. Then, each partition is indexed using the MBB tree where each node is embedded with Membership Vector (MV). MV is a binary string depicting the presence of a keyword in a document with bit one. The partitioned indexing structure supports parallel search, and the proposed search scheme achieves sublinear search time. Also, the proposed PECS supports dynamic update with minimum overhead. Our contributions are summarized as follows:

1. Proposed and implemented a novel Privacy enhanced conjunctive ranked search, PECS, over the encrypted data.
2. Proposed and implemented a novel indexing scheme, TPIS, which supports parallel search and dynamic update on the document collection outsourced.
3. PECS ensures keyword, query, and index confidentiality and privacy under known background threat model.
4. Sublinear search efficiency is achieved using the proposed search framework performing a search in parallel across the MBB trees in TPIS.

The organization of the paper is as follows. Section 2 provides an overview of related works. Section 3 provides problem formulation giving details of the system model. Section 4 illustrates the proposed search scheme and associated algorithms necessary to perform indexing and search to achieve security and privacy. Section 5 provides a detailed theoretical analysis of PECS. Section 6 focuses on experimental analysis and verification of the efficiency of the proposed scheme. Finally, Section 7 summarizes our findings and offers valuable insight into potential future improvements.

2. Related works

Often the data are outsourced in the form of encrypted files due to security reasons which reduce the data usability and introduces many challenges in the retrieval and processing of outsourced information. Searchable encryption schemes enable the user to perform the keyword search over the encrypted domain on a remote cloud server. There are two types of searchable encryption in the literature: using public key cryptosystem and using private key cryptosystem. Abundant works have been proposed in both schemes.

The first searchable encryption using symmetric cryptosystem is proposed by Song et al. [9]. The proposed system supports single keyword search without an index and the search time is linearly proportional to the cardinality of the document collection set. Later, many proposals were suggested in the literature improving the search efficiency by introducing index keyword for faster search. Goh et al. proposed a formal searchable encryption scheme and introduced encrypted indexing using Bloom filter which improved the search efficiency, but the time cost is $O(n)$ [13]. A searchable scheme independent of the underlying encryption algorithm is proposed [14]. An optimal and secure SSE schemes are proposed in the work [15], the where authors provided comprehensive proofs regarding chosen keyword attack and adaptively chosen keyword attack. In these works, the order preserving techniques are utilized to provide ranking information and they provide better search results based on the underlying inverted index built upon the keyword frequency information. However, most of these works supported single keyword boolean search with fixed keyword dictionary and may not scale well with the size of the data outsourced.

Ranked search schemes evaluate the query based on a relevant score function and return Top-k documents. Wang et al. [16] proposed a ranked search based on the relevance scores of keywords. Sun et al. [17] proposed a verifiable multi keyword search based on keyword relevance score. The authors constructed a searchable index tree using vector space model. A more efficient multi-keyword ranked search (MRSE) is proposed by Cao et al. [18] in which documents and keywords are represented as vectors. Here, authors improved the search accuracy by returning documents according to the number of matching keywords using coordinate matching. A dynamic SE for a very large database is recommended in [19]. Xia et al. [20] proposed a secure and dynamic multi-keyword ranked search over encrypted data. Authors improved the search efficiency by keyword balanced binary tree indexing but do not achieve sublinear search efficiency with increase in document size. Cengiz et al. suggested a MinHash based privacy preserving multi-keyword search over encrypted data [21]. Authors proposed a novel obfuscation method to hide search access patterns. The search framework includes two servers namely file server and a search server but fails to achieve update operation due to the hiding of access pattern. A personalized search is proposed by Zhangjie et al. by formulating user search history using ontology [22]. Xiaofeng et al. [23] recommended a multi-keyword top-k search and defined random traversal algorithm using document replication. However, this scheme introduces more communication and storage overhead due to replication. Recently, many modifications to multi-keyword search are proposed in the literature which shows the demand for efficient practical search scheme [24–30]. Although the above multi-keyword search schemes provide top-k search results by the number of matched keywords, more accurate ranking is not considered and cloud server traverses every index path for each search request.

Boneh et al. [31] proposed the first public key cryptosystem based SE scheme. The proposed scheme supports any user to encrypt the data with the public key, but only authorized users are allowed to create the trapdoor using the private key. In the public key cryptosystem domain, many works have been proposed for conjunctive search and subset search [32–34]. However, these search schemes induce heavy computational overhead to achieve search efficiency with increase in the volume of data. Golle et al. [32] proposed a secure conjunctive search framework. Authors used two protocols based on bilinear pairings to achieve secure search. A verifiable conjunctive keyword search over mobile e-health cloud scheme is proposed [35] in a shared multi-owner setting. Recent works on searchable encryption focus on the expressiveness of the query term [36]. Verifiable conjunctive search schemes in [37,38] induce overhead due to verification and delays the query response. Conjunctive search schemes discussed here do not support document scaling due to the heavy computational overhead of bilinear map.

The idea of multi-keyword ranked search inspired the research community to come up with variants to enhance the search experience. Although the above works provide better query efficiency with static keyword dictionary, these schemes require rebuilding the entire index to perform both keyword and document update operations. Practically, the data owner wants the support for document insertion and deletion operations. Moreover, the poor query efficiency with document scaling is a problem with most of the real-time data outsourcing applications. The SE schemes based on public key cryptosystem also suffers from keyword update and scalability issue because of the computational overhead for index generation. In the proposed work, we address the issue of update operation and query efficiency with document scaling.

In this paper, we propose PECS which not only supports multi-keyword search but also achieves sublinear search efficiency with TPIS indexing by performing parallel search and filtering search paths. In addition, with the partitioned indexing, PECS provide scalability with the number of documents and owner defined keyword weighting approach preventing the statistical attack. The proposed conjunctive search framework using TPIS adopts the secure kNN [39] and achieves better index creation time cost, search efficiency and update time cost thereby meeting the privacy goals.

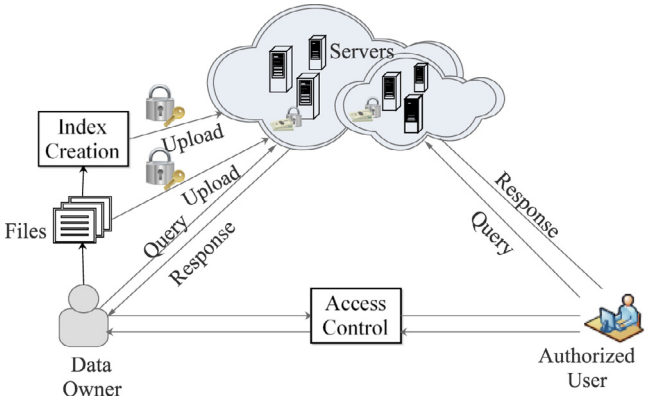

Fig. 1. System model [18].

3. Problem formulation

3.1. System model

The proposed PECS scheme considers a data outsourcing environment consisting of three primary entities namely: the data owner (DO), data users and the cloud data service provider (CDSP) as depicted in Fig. 1.

Each entity plays a different and unique role in the data outsourcing environment. The DO has a set of documents $F = \{f_1, f_2, ..., f_n\}$ to be outsourced to the cloud to provide data management solutions for its clients. For example, it may be a health organization where huge records (mostly text and image data) are generated daily and will seek the help of cloud data services to provide sufficient storage and computing power on demand. However, these type of sensitive data outsourcing needs most robust security methods to ensure confidentiality and privacy over the data outsourced since the DO have limited/no control over the outsourced data.

To provide access to other legitimate data users, the DO constructs a searchable index using the defined keywords over the document collection, and the encrypted index is also outsourced to the CDSP. A user is an authorized entity including the DO who may access the uploaded information later after acquiring the proper access information. To initiate a search over the encrypted data, the data user obtains the corresponding trapdoor through access control mechanism from the DO. A CDSP is capable of providing sufficient storage and computation resources as service on demand to the users. On receiving the intended trapdoor information along with encrypted query term from a data user, the CDSP initiates a search process and returns a list of all relevant documents. To improve the search efficiency, we propose a novel indexing structure TPIS. PECS provide an efficient parallel search scheme and Table 1 lists the major notations used in this paper.

Table 1
Notations.

Notation	Description
F	The plaintext document collection of n documents, $F = \{f_1, f_2, ..., f_n\}$
W_i	The set of keywords associated with file f_i in F, $W_i = w_1, w_2, ..., w_m$
F_{ID}	Unique file identifier
C	The set of encrypted files, $C = C_1, C_2, ..., C_n$
I	The encrypted index built for F
W	The universal set of keywords defined over F and $W = W_1, W_2, ..., W_n$
Q_w	The subset of keywords in the query, where $w \subset W$
TD_w	Trapdoor generated for the query Q_w
D_w	The set of documents returned for the query Q_w
P	The set of p partitions as $P = p_1, p_2, ..., P_p$, where $p = \log n$

3.2. Threat model

In this work, we assume that both DO and data users are trusted entities and consider the CDSP as a "honest-but-curious" entity as in most of the secure cloud data search frameworks [40]. A CDSP is honest in satisfying the user request by following the service protocols. However, it can silently obtain additional information such as access pattern, index locations accessed, query keywords used and the number of messages exchanged which may lead to security attacks on the outsourced data. The proposed work considers two threat models namely known ciphertext model and known background model [41].

3.2.1. Known ciphertext model

The CDSP has access to encrypted document collection, encrypted search index, and encrypted query term. We assume that in this model the CDSP does not have any additional information to obtain the original data.

3.2.2. Known background model

In the known background model, the adversary has more background knowledge regarding the document collection by performing statistical analysis on a similar dataset. For example, the CDSP may have more statistical information regarding the distribution of term frequencies of keywords. The known background model is stronger than known ciphertext model.

3.3. Preliminaries

We use keyword priority weighting to denote the relevance score between keywords and the document. Most of the existing works use TF-IDF to express the relevance score. But, the TF-IDF may leak statistical information regarding the distribution of keywords in the document and may not suitable for sensitive information like health and finance data. In the proposed work, the keywords are defined by the DO, and each keyword has a weight factor in the interval (0, 1) to express the importance. For a given query with t keywords, the document score is determined using Eq. (1).

$$Score(f_{id}) = \left\{ \sum_{n=1}^{t} k_n \middle| k_n \in f_{id} \right\} \tag{1}$$

The objective of multi-keyword top-k search with keyword priority weight dictionary is to retrieve the top-k similar documents from the document collection F outsourced to the CDSP. Formally the search is defined as follows

Definition 1. Given a query Q with t keywords, and document collection F, the multi-keyword search determines the relevance score of each document using Eq. (1) and returns the top-k relevant documents.

For example, consider a document collection to upload, $F = \{f_1, f_2, f_3, f_4, f_5\}$, and each document is represented as a vector using keyword priority weight as illustrated in Table 2. The universe of keyword space has a total of eight keywords and the importance of a keyword is shown by assigning a higher value. For the query vector $Q_i = (1, 1, 0, 0, 1, 0, 0, 0)$, the search procedure determines the

Table 2
Document vector representation.

	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8
f_1	0.3	0.5	0.6	0	0	0.9	0.2	0
f_2	0.6	0	0.3	0.2	0	0.2	0.8	0
f_3	0	0.3	0	0.7	0.5	0	0	0
f_4	0.8	0.3	0	0	0	0	0.2	0.6
f_5	0.5	0.2	0	0	0	0	0.4	0.7

relevance score of each document using Eq. (1) and is {(f_1, 0.8), (f_2, 0.6), (f_3, 0.8), (f_4, 1.1), (f_5, 0.7)}. For top three search, it returns the documents f_4, f_1, and f_3.

3.4. Design goals

The main design goals of the proposed search framework PECS are: (1) To propose a novel and scalable index structure to improve search efficiency; (2) multi-keyword top-k search over encrypted data supporting parallel search; and (3) to ensure index and query privacy.

1. Scalable index structure: Data outsourcing scenario usually deals with a large number of documents, and it needs scaling up with demand. To support scaling up of document size and efficient search process, a novel index structure TPIS is proposed and implemented. TPIS supports parallel search and improves the search efficiency.
2. Multi-keyword top-k search: The proposed search framework should support multi-keyword search over encrypted data. In the case of sensitive information, the retrieval of relevant documents to the query is desired. Therefore, a novel relevance ranking method is proposed and implemented in this work which enhances the search accuracy. The importance of keywords in a file and user input are considered for the calculation of relevance score.
3. Privacy preserving: The proposed search framework PECS should provide both index privacy, query privacy, and keyword privacy. The special structure of the index proposed TPIS protects information regarding the correlation between keywords and indexes. The objective is to prevent the CDSP from guessing the relationship between documents and keywords using the index. A secure and random trapdoor value is generated each time to prevent query privacy. Random trapdoors are generated to provide trapdoor unlinkability feature.

3.4.1. Membership balanced binary tree (MBB)

The balanced binary tree is widely used in optimization problems [20]. The membership binary tree is a balanced binary tree in which each node has an additional bit vector called membership vector (MV). The data records are stored at the leaf nodes of the tree similar to B +Tree. However, in comparison to the binary tree, the internal nodes store a vector consisting of membership bits for the keywords and pointers to both left child and right child. Therefore, the modified tree is called membership balanced binary tree. Algorithm 2 illustrates the tree creation in detail. The node structure is defined as follows:

Definition 2. *Node* = $\langle lchild, MV, rchild \rangle$, bit 1 in i^{th} position of MV indicates the presence of keyword w_i in documents indexed by the subtree rooted at the respective node, whereas bit 0 indicates the absence of keyword w_i.

4. Proposed scheme PECS

In this section, we first describe the proposed novel index structure TPIS which is constructed using the partition table and MBB tree. Using TPIS two secured multi-keyword top-k search schemes are proposed against the two threat models.

4.1. TPIS Construction

The document collection F is randomly mapped into $m = \log n$ partitions and each partition is organized using MBB tree. The storage organization is shown in Fig. 2. The partition table can be viewed as a forest. It consists of a collection of disjoint MBB trees. The partitioned index space allows us to perform the search in parallel and enhances the search efficiency. The DO constructs the searchable index I before outsourcing to the cloud. For each document f_i in F, the DO defines a set of keywords. In the scenario of sensitive information outsourcing, the use of most frequently occurring terms as keywords may lead to information leakage. Therefore, TPIS uses user-defined words which describes the document better and is useful in the case of sensitive files. Then a universal keyword dictionary is formed by taking the union of keyword sets of all documents in the collection F, $W = \{W_1 \cup W_2 \cup ...\cup W_n\}$. The original collection of documents is divided randomly into different groups, $P = \{p_1, p_2, ..., p_m\}$. Each partition p_i is then indexed by an MBB tree. For each document group, the DO constructs an MBB tree with the following properties.

- Data files are stored in the leaf nodes.
- Each internal node stores links to left child, right child, and a membership vector MV.
- The length of MV is same as the keyword dictionary defined over that partition. Each bit denotes whether the corresponding keyword exists or not in the document stored in the respective leaf node.
- The MV assigned to the leaf node is called weighted MV since the membership bit now shows the relevance or weight of that keyword in that document.

The TPIS index construction involves two major steps:

1. Random mapping is done to logically partition the original document collection F. Now, we have partitioned document sets as $P = P_1, P_2, ..., P_k$. The random mapping introduces more entropy.
2. Then, for each partition create a balanced binary tree with membership vector embedded in the node itself.

The major advantages of partitioning the entire document collection into subpartitions are twofold. First, the partitioning of original document space into subspaces based on random mapping improves search efficiency by enabling parallel search capability. Second, it reduces the keyword vector length by keeping a partition keyword vector instead of global keyword vector for F. The details of partition process are illustrated in Algorithm 1. Each partition keyword set is created by taking the union of keywords of member documents of that partition.

TPIS index creation involves the construction of MBB tree for each partition. An example MBB tree is shown in Fig. 3.

The construction of the tree follows a bottom-up approach, and the resulting tree will be balanced. For each node in MBB tree, the proposed algorithm embeds a membership vector which is stored as a bit string. Each bit in the vector indicates the presence or absence of the keyword in the document corresponding to the partition keyword space. Suppose that partition keyword space $W_{P_i} = w_1, w_2, w_3, ..., w_{10}$, then a document having keywords w_1, w_2, w_3, w_9 have membership vector $MV = 1110000010$. A leaf node is introduced for every document in the partition with document identifier and membership vector. Then, parent nodes are constructed by pairing leaf nodes from left to right. In the event of an odd number of nodes at any level, the last node is paired with the parent of previous two nodes. For any internal node, the membership vector is created by Boolean OR operation of membership vectors of child nodes. The details of the MBB tree creation is shown in Algorithm 2.

4.1.1. Query construction

Given the query vector Q with t keywords, a group of query vector is created by the DO. The proposed indexing structure enables parallel search across m partitions. Therefore, each query vector is transformed into m query vectors of random length which introduces randomness in access pattern and enhances query unlinkability. Query group is identified as $\{Q_1, Q_2, ..., Q_m\}$, where Q_i denotes query intended for partition p_i with length d_i. The k^{th} dimension of query vector Q_i corresponds to the existence of keyword in the query. Finally, the user submits the query vector to the CDSP.

Input: Document Collection $F = \{f_1, f_2, \ldots, f_n\}$
Output: Storage Index I
/* Initialize document keyword sets */
$W_1, W_2, W_3, \ldots, W_n \longleftarrow$ NULL;
Initialize universal set of keyword $W \longleftarrow$ NULL ;
Initialize partition tables $P = P_1, P_2, \ldots, P_P$ as NULL;
foreach *(f_i in F)* **do**
    extract keyword, w;
    Add w to the respective keyword set W_i;
    $W_i \leftarrow W_i \cup w$;
end
$W \leftarrow W_1 \cup W_2 \cup \ldots \cup W_n$;
foreach *(f_i in F)* **do**
    $j \leftarrow random[1..p]$;
    put f_i in P_j;
    $W_{P_j} = W_{P_j} \cup W_i$;
end
foreach *(P_i in P)* **do**
    $root \leftarrow CreateMBB(P_i)$;
    $I[i] \leftarrow root$;
end
Return I;

Algorithm 1. BuildIndex

Input: Partition P_i, weighted keyword set KW_{P_i}
Output: Root pointer of the MBB tree created

foreach *(f_i in P_i)* **do**
  Create node n;
  $n \leftarrow$ lchild = NULL, $n \leftarrow$ rchild = NULL, $n \leftarrow$ data = f_{ID};
  foreach *kw_j in f_W* **do**
    $n \leftarrow MV_j = weight(kw_j)$;
  end
  Insert node n in the ActiveNodeList ;
end
while *|ActiveNodeList| > 1* **do**
  if *Number of nodes in ActiveNodeList is even* **then**
    foreach *(pair of nodes (n_1, n_2) in ActiveNodeList)* **do**
      remove (n_1, n_2) from ActiveNodeList;
      Create parent node, n;
      $n \leftarrow$ lchild = n_1, $n \leftarrow$ rchild = n_2 ;
      $n \leftarrow$ MV = $n_1 \leftarrow MV$ OR $n_1 \leftarrow MV$;
      Insert n into TempList;
    end
  else
    Perform the same steps in the IF case;
    Insert last node (2h+1) into TempList;
  end
  $ActiveNodeList \leftarrow TempList$;
  $TempList \leftarrow NULL$;
end
Return RootPtr;

Algorithm 2. CreateMBB.

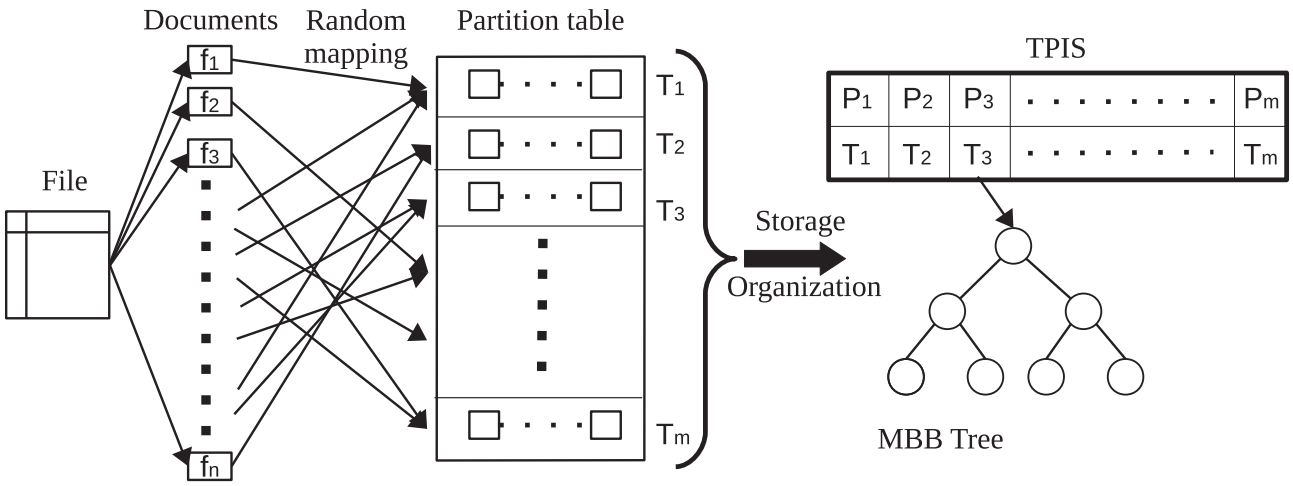

Fig. 2. Proposed storage structure, tree-based partitioned index tree.

4.1.2. Search process

The details of the search process in CDSP is shown in Algorithm 3. The proposed indexing TPIS organized document collection as distinct but connected MBB trees. The search can also be initiated in parallel mode using multiple threads. For instance consider the example MBB tree in Fig. 3 and query vector, $Q = w_1 \wedge w_2 \wedge w_5 \wedge w_6$. Two threads will be initiated from the root node 1. The first thread initiates a search at node 2 and second at node 3. The first thread terminates after searching node 3 and 4 with no matching answer. The second thread will continue the search similarly and returns the document identifier stored at last right leaf node as the answer. The search in MBB tree follows a path similar to post order traversal and the process is shown in Algorithm 3.

4.2. Basic PECS

The relevance of keywords for each document is ensured by enabling the data owner to have an individual keyword set. The baseline for process of key generation and index creation is adapted from MRSE [40]. Each partition of TPIS is going have its own keyword dictionary based on the member documents. The process of encrypting the document collection and index I is explained below:

$SK = SetUp(1^l)$. Taking a security parameter l as input, the data owner outputs secret key as $SK_i = (S_i, M_{i1}, M_{i2})$ for each partition in the index space. S_i is a randomly generated l-bit vector where $l = |W_i|$. M_{i1} and M_{i2} are two invertible matrices of dimension (lxl). Finally, SK is generated as $SK = (SK_1, SK_2, \ldots, SK_m)$. The DO shares ($SK$, K) with other authenticated data users where K is the symmetric key used to encrypt the documents outsourced to the cloud.

$I = BuildIndex(F, S\ K)$. Using the TPIS construction algorithm, the unencrypted index is created. An index vector of dimension m is created for each document f_i in P_i. Each node (except leaf node) in the MBB tree is the membership vector which looks like a random bit string. For the leaf node, the random bit string is replaced with the weighted membership vector. Each node index is divided into two vectors as (DN^1, DN^2) using the vector S. If k^{th} bit of S is 0 then $DN^1[k] = DN^2[k] = DN[k]$ otherwise k^{th} bit of one of the vector N^1 or N^2 is set to bit 1 and the other is set to bit 0 which satisfies the sum property specified in [40]. The encrypted index for each MBB tree is created as is $I_i = (M_{i1}^T DN^1, M_{i2}^T DN^2)$. The final encrypted index $I = I_1, I_2, \ldots, I_m$ is generated.

$TD \leftarrow Trapdoor(Q, S\ K)$. Given the keywords in the query term Q, the query group is generated as $Q = (Q_1, Q_2, \ldots, Q_m)$. For each query vector Q_i in Q, the query membership vector Q_{iq} of dimension m is created showing the presence (bit = 1) and absence (bit = 0) of keywords. The query vector Q_{iq} is divided into Q_{iq}^1 and Q_{iq}^2 using S_i as the splitting function. The encrypted query vector is constructed as $\overrightarrow{Q_{iq}} = (M_{i1}^{-1}Q_{iq}^1, M_{i2}^{-1}Q_{iq}^2)$. The final trapdoor is generated as $TD = (\overrightarrow{Q_{i1}}, \overrightarrow{Q_{i2}}, \ldots, \overrightarrow{Q_{im}})$ and is send to the CDSP.

$D \leftarrow Query(\overrightarrow{Q_{iq}}, I)$. Query evaluation involves membership checking and relevance score calculation. The relevance score is determined as follows:

$$\begin{aligned} I.\ TD &= (M_{i1}^T DN^1.\ M_{i1}^{-1}Q_{iq}^1) + (M_{i2}^T DN^2.\ M_{i2}^{-1}Q_{iq}^2) \\ &= ((M_{i1}^T DN^1)^T.\ M_{i1}^{-1}Q_{iq}^1) + ((M_{i2}^T DN^2)^T.\ M_{i2}^{-1}Q_{iq}^2) \\ &= DN^{1T}M_{i1}M_{i1}^{-1}Q_{iq}^1 + DN^{2T}M_{i2}M_{i2}^{-1}Q_{iq}^2 \\ &= DN^1.\ Q_{iq}^1 + DN^2.\ Q_{iq}^2 \\ &= (DN.\ Q_q) \\ &= RS(Document - at - node - DN) \end{aligned} \quad (2)$$

4.2.1. Index confidentiality and query confidentiality

The index vector I and the trapdoor TD are simple random bit string from which the CDSP cannot generate the original index value and query vector without the secret key S. The matrices M_1 and M_2 are Gaussian random matrices, and the attacker cannot determine these matrices from the ciphertext [20]. Moreover, instead of a single secret key as in most of the proposed scheme, the proposed model uses m secret keys which increases the randomness in key guessing process. Hence, the proposed secure indexing scheme TPIS is resilient against known ciphertext attack and ensures both index and query confidentiality.

4.2.2. Query unlinkability

Initially the query vector is transformed as query group as $Q = (Q_1, Q_2, \ldots, Q_m)$. The trapdoor for each query vector Q_i is generated by the random splitting of Q_i using the respective secret key vector S_i. For the same set of keywords, the random split process generates different trapdoors making query unlinkability difficult. Since all partitions are always searched, the attacker cannot deduce any information regarding frequently accessed partition. However, the CDSP may observe the visited path and relevance score of the search process.

4.2.3. Keyword privacy

The encrypted index and query vector protects the confidentiality of keywords. The search process uses only the inner product operation protecting the keyword privacy. Therefore, the proposed scheme provides keyword privacy in the known ciphertext model. However, in the known background model, the attacker is assumed to have more knowledge regarding the keyword distribution from statistical analysis. The proposed model prevents statistical attacks to an extent by using

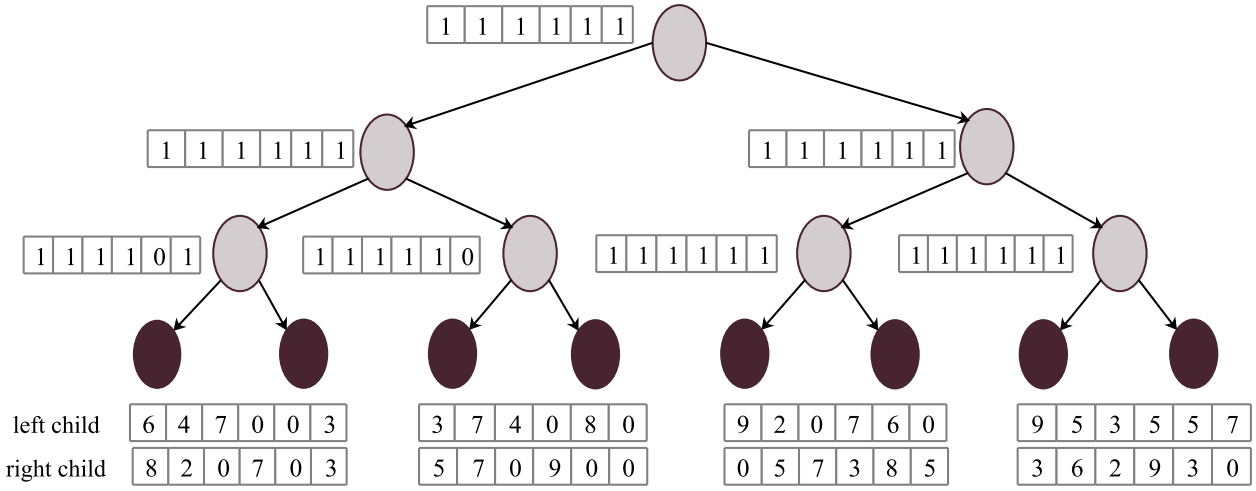

Fig. 3. Membership tree construction.

defined keywords in the index creation rather than term frequency. But, access pattern may be deduced which may help the attacker to launch a statistical attack. Specifically, when the number of keywords in the query is one or two, then the CDSP can observe the relevance score. Therefore, the basic should be modified to make stronger in the known background model.

4.3. Enhanced PECS

The security analysis shows that the basic search scheme should be modified to provide keyword privacy in the known background model. The information leakage is due to the calculation of relevance score which is equal to the original relevance score of the document. Therefore, we introduce varying level of randomness in each partition of the index space for each query request. This introduces maximum randomness in relevance score calculation process which prevents the statistical attack. Dummy keywords are added to randomize the distribution of keyword usage and relevance score calculation as given in [18]. The enhanced search scheme EPECS differs from the basic scheme BPECS in the following aspects:

- Setup: Each secret key S_i is extended to the dimension of $(m + r)$ where r is the number of dummy keywords introduced in the respective dimension. Also, the Gaussian matrices M_1 and M_2 are extended to the dimension $(m + r)(m + r)$.
- BuildIndex: Each node vector of the unencrypted tree is extended to the dimension of $(m + r)$ to incorporate the set of dummy keywords. Among the r dummy keywords, a certain number of keyword vector bit positions are set to the weight value. The splitting process is same as in basic scheme. The weight of dummy keywords is set to least value so that the inclusion of dummy keywords does not affect the final relevance score.
- Trapdoor: Each query vector in the query group is extended to (m_i, r_i) respective to that partition and its dummy keyword set.
- Relevance Score: The final relevance score includes actual score plus the dummy score which is ignorable as we have assigned least weight value to dummy keywords.

4.3.1. Security analysis

As with the basic scheme, the enhanced scheme is also analyzed concerning the security and privacy goals as follows:

4.3.1.1. Index confidentiality and query confidentiality. The enhanced scheme EPECS introduces more randomness in the secret vector S_i and Gaussian matrices M_1 and M_2 by adding a random number of dummy keywords to each partition. The inclusion of a random number of keywords in each partition makes it almost impossible to guess either the secret vector or matrices. Hence, the proposed EPECS method enhances the confidentiality of both index and query vector in the known background model.

4.3.1.2. Query unlinkability. The inclusion of a dynamic number of keywords for each query and random splitting has introduced more randomness in the relevance score distribution. For the same query, the EPECS has produced different relevance score since each time the query vector includes a varying number of dummy keywords. This makes the attacker find the link between query vectors. Moreover, the data user can control the dummy score value by adjusting the selection value of dummy keywords in the query vector. Therefore, the proposed EPECS ensures query unlinkability in the known background model.

4.3.1.3. Keyword privacy. The keyword privacy is improved in EPECS by including a random number of dummy keywords in the query vector to disturb the distribution of both keyword usage frequency and relevance score. We assume that each partition uses a mutually exclusive set of dummy keywords. Suppose that there are 2^{ω} different choices for the dummy keywords for each tree node in the index, then the probability of having the same set of dummy keywords for any two nodes is $\frac{1}{2^{\omega}}$. The random numbers E_j, (j = 1 to ω) follow uniform distribution with $U(\mu` - \delta, \mu` + \delta)$. According to the central limit theorem $E_V = \sum_{j=1}^{\omega} E_j$ follow normal distribution with mean μ and variance σ^2. The number of different dummy keywords is not greater than $\binom{r}{r`}$ which is maximized when $\frac{r}{r`} = 2$. In this case $r = 2\omega$ and $r` = \omega$. This indicates that every data vector should include at least 2ω dummy entries and every query vector will randomly select half of the dummy entries. Here, ω is the system parameter which acts as the compromising factor between efficiency and privacy.

5. Theoretical analysis

To get a more accurate result and satisfactory search experience, a multi-keyword search query is needed. The creation of keyword dictionary for each document and global keyword dictionary is based on term frequency in MRSE and VCSE which is prone to statistical attack with strong background information. Moreover, as with sensitive information such as medical records and financial records, the frequent words would reveal a hint towards the content of the document. Therefore, PECS allows the DO to define customized document dictionary with random keyword weight which will help to prune the search domain with a multi-keyword search.

Input: MBB tree pointer, Root, and Query Vector Q
Output: DList consists of all matched document identifiers
1 $V = Root \rightarrow MV$;
2 **if** *all* $V[i] = 1$ *with respect to* Q **then**
3 Find relevance score of this document and query vector Q;
4 $Score(f_{id}) = \left\{\sum_{n=1}^{t} k_n | k_n \in f_{id}\right\}$;
5 Insert $Root \rightarrow FID$ into Dlist if the score is greater than threshold;
6 $Root = Root \rightarrow lchild$;
7 $Root = Root \rightarrow rchild$;
8 **else**
9 Return;
10 **end**
11 Return Dlist

Algorithm 3. SearchMBB.

Table 3
Functional analysis.

Metric	MRSE [18]	VCSE [37]	Proposed-PECS
Multi-keyword	✓	✓	✓
Relevance Score	✓	✓	✓
Extension of Query term	✓	✓	✓
Keyword Weight	✓	No	✓
update	No	No	✓
Scalable index	No	No	✓

In this section we provide a functional analysis of PECS which is also compared with the other two relevant schemes, MRSE [18] and Verifiable Conjunctive Search scheme over Encrypted data (VCSE) [37]. MRSE has provided the first practically efficient multi-keyword search over encrypted data by coordinate matching. VCSE focuses on providing a verifiable multi-keyword search preventing keyword guess attack. The functional analysis is shown in Table 3 against the significant functional metrics involved in the index creation and query processing. The update of keyword dictionary is an expensive operation in both MRSE and VCSE since the whole index has to be modified. However, the partitioned index structure TPIS permits PECS to perform update operation efficiently since each partition is maintaining a local dictionary. Only the sub-index of TPIS needs modification during an update operation which significantly reduces the cost of updation as compared to that of MRSE and VCSE.

The partitioned index structure TPIS imparts document scaling effectively. Each partition of the index is represented by MBB tree structure which presents satisfactory addition or deletion of keywords and documents. For a mass addition or removal of documents, TPIS needs to either add or remove a partition with less cost. The scalability of MRSE and VCSE is limited since they require a modification of the entire index structure with every addition or removal operation. Therefore, compared to MRSE and VCSE, PECS presents a scalable index structure necessitated by practical scenario.

We also evaluate the performance of PECS in terms of theoretical computational complexity and its actual performance by experimental analysis. To perform the theoretical computational complexity analysis, we define the following parameters: T_S, T_{BI}, T_{TD} and T_{QE} as the time cost to generate setup parameters, build the index, trapdoor generation and query evaluation respectively. The comparison results in Table 4 shows the theoretical performance dominance of PECS as compared to that of MRSE and VCSE.

The conjunctive search algorithm in VCSE uses bilinear pairing which includes exponentiation and pairing operations. Let E_1 denote exponentiation in group G_1, E_2 denote exponentiation in group G_2 and P represent the number of bilinear pairing operations required to perform the key generation, index creation, trapdoor generation and query evaluation. MRSE and PECS use secure kNN approach which includes a symmetric encryption cost of SE_t which is obviously less than bilinear pairing operation. The setup time of PECS is almost same as MRSE as it creates secret random vector and two invertible Gaussian matrices. In PECS the key is generated for each partition, but with reduced matrix dimension. The setup time of VCSE is larger as compared to both PECS and MRSE since it includes exponentiation operation required to generate both public and private key pair.

Table 4
Theoretical computational analysis.

Scheme	T_S	T_{BI}	T_{TD}	T_{QE}
MRSE [18]	$3(n+1)$	$SE_t + n^2.m$	$t.n^2$	$n^2.m.t$
VCSE [37]	$n+2E_1+2E_2$	$E_2 + n(2E_1 + E_2)$	$E_1 + E_2$	$4P+3E_1+2E_2$
Proposed-PECS	$\log n(3(\log n + 1))$	$SE_t + \log n.\log m.m^2$	$m^2.t$	$\log n.m^2.t$

The index creation cost includes the encryption of document collection F and creating individual document index with split operation and dictionary creation. The individual document index creation and dictionary creation time of PECS are less as compared to that of MRSE since PECS uses partitioned sub-index structure by TPIS which leads to reduced matrix dimension and dictionary size. Each partition in TPIS has utmost $\log n$ nodes and takes $O(p)$ to create index vector for each node, p is the local dictionary size. Node encryption includes node vector splitting and two matrix multiplication of dimension $(p \times p)$. For each node vector splitting process takes $O(p)$ and two multiplications of complexity $O(p^2)$. Hence, we conclude that the index creation cost of PECS is of the order $O(\log np^2)$ for each partition. The complexity is proportional to the reduced dictionary size,p, in PECS whereas in MRSE it is proportional to the size of global keyword dictionary $m = |W|$. Therefore in MRSE, the index creation requires the multiplication of order $(m \times m)$. The indexes of PECS is always encrypted by the local dictionary size thereby achieving better index creation cost. The sub-index structure TPIS enables parallel query execution making query evaluation faster in PECS. With VCSE the trapdoor generation and query evaluation cost are larger due to heavy exponentiation and pairing operations involved. The subsequent section details the actual performance analysis of PECS.

The trapdoor generation time depends on the size of the keyword dictionary. In PECS the trapdoor is divided into $\log n$ parts and each part maintains a sub-dictionary of size m. Therefore, the complexity of trapdoor generation in PECS is of the order of $O(m^2.t)$ where t is the number of terms in the query expression and two multiplication of order m. With MRSE the trapdoor generation cost is of the order of dictionary size which is $O(n^2.t)$. Since the trapdoor generation cost is depending on the dictionary size, EPECS outperforms MRSE with reduced dictionary size. The trapdoor generation in VCSE is based on the private key of a user which involves exponentiation operation which is proportional to the number of keywords in the query expression. Hence, PECS shows better trapdoor generation cost as compared to that of MRSE and VCSE. The query evaluation efficiency also depends on the document size, dictionary size and the number of terms in the query expression. PECS has better evaluation cost than MRSE because of the partitioned index and reduced dictionary size. VCSE consumes more time for query evaluation due to the pairing of bilinear map. The subsequent section details the actual performance analysis of PECS.

6. Experimental analysis

The proposed search model is implemented using Java language in Windows 8 operating system and tested its efficiency on a real-world document collection: the Request for Comments(RFC).[1] The test includes (1) the search precision on different privacy level, and (2) the efficiency of index construction, trapdoor generation, search, and update. The experimental results are obtained with an Intel(R) Core(TM) i7-4790 Processor (3.60 GHz) which supports eight parallel threads.

6.1. Precision and privacy

As discussed in Section 2, to decrease the size of the index and improve the query efficiency PECS employ partitioned index structure TPIS. To evaluate the precision and rank privacy, the proposed model uses the same definition from [18]. Precision is defined as $P_k = k'/k$ where k' is the number of actual top-k documents in the retrieved list, and k is the total number of documents in the retrieved list. To protect keyword privacy in the query evaluations, EPECS add a random number, E_j, of dummy keywords to each query expression which can affect the precision of result set. The number E_j follows normal distribution, $N(\mu, \sigma^2)$. The precision is mostly depending on the number of dummy documents and the dummy score $\Sigma_v Ej$, which can be balanced by choosing the proper variance σ. The results are shown in Fig. 4(a) indicates that the proposed method achieves higher precision with comparatively better variance value since the proposed index structure itself has a certain degree of randomness due to the partitions and group query formation. The uncertainty in the relevance score calculation is quantified using rank privacy as follows:

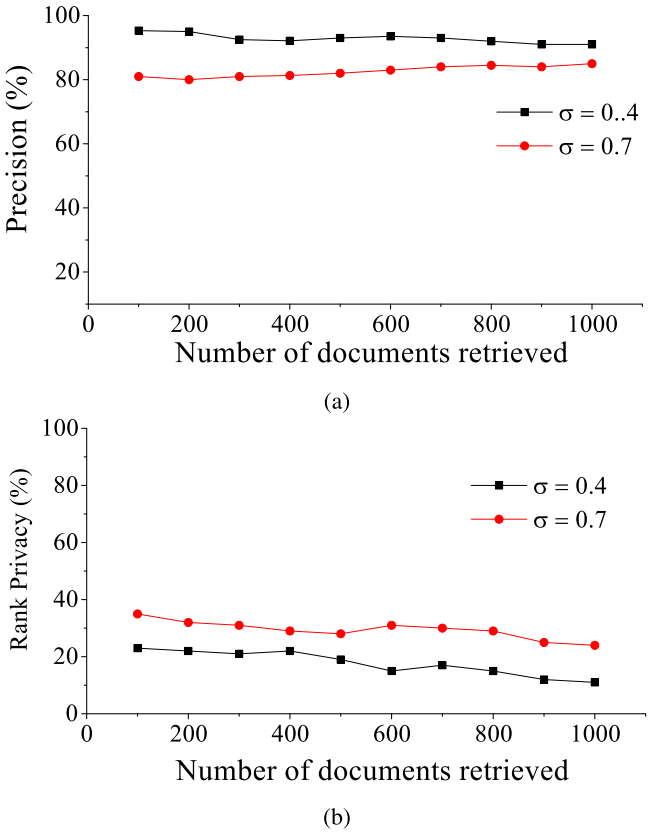

Fig. 4. Showing the effect of dummy keywords on variance of the distribution and its impact on the precision and rank privacy of the retrieved documents. (a) Precision, (b) Rank Privacy.

$$P'_k = \sum |rn_i - rn'_i|/k^2 \tag{3}$$

The actual rank of the document f_i in the retrieved list is rn'_i and obtained rank is rn_i. Better rank privacy is an indication of more security as shown in Fig. 4(b) and variance σ acts as the balancing factor. By choosing proper variance value, the adequate tradeoff between precision and privacy can be achieved as illustrated in Fig. 4. To make an optimal tradeoff, we need to know the maximum precision achievable for a given level of rank privacy and vice versa. The optimal value for σ depends on the choice of precision and rank privacy which is an optimization problem. The achievable query relevance can be stated as

$$Acc(P_k, P'_k) = W_1.\ P_k + W_2.\ P'_k \quad where \quad W_1 + W_2 = 1 \tag{4}$$

Here, the weights W_1 and W_2 present the preferences between precision and rank privacy. The preference is fixed for a particular choice by tuning the parameter σ.

6.2. Index tree construction

The process of TPIS index construction for document collection F includes two main steps: (1) Random mapping into partitions, and (2) constructing and encrypting the MBB index tree. The index structure is constructed by a random mapping function followed by MBB tree construction. The random mapping involves linear time complexity and

[1] https://www.rfc-editor.org/retrieve/.

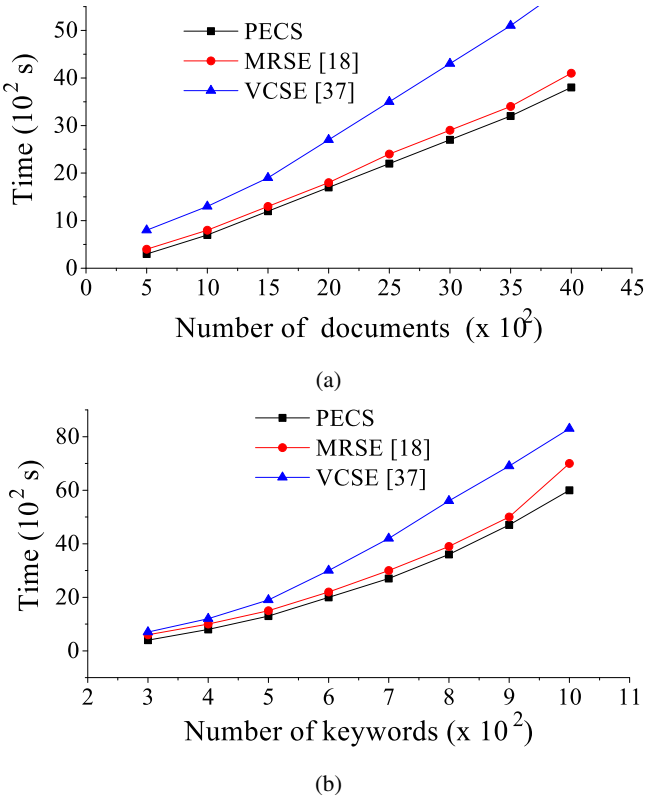

Fig. 5. Time cost for TPIS index construction. (a) Time Cost for TPIS Index Construction with Fixed Keyword Dictionary, W = 5000, (b) Time cost for TPIS index construction with fixed document size, $n = 1000$.

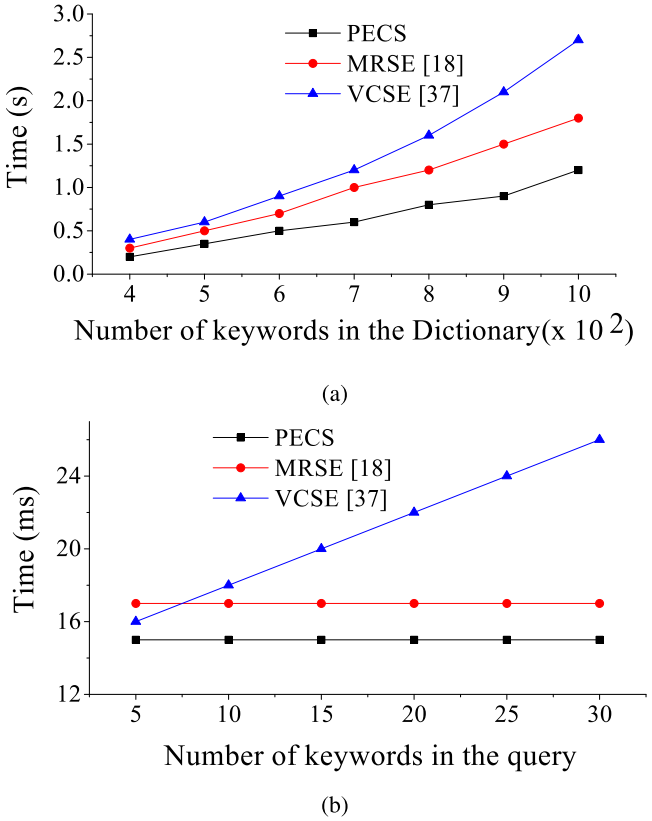

Fig. 6. Time cost involved in trapdoor generation for a user request. (a) Time cost involved in trapdoor generation with varying the keyword dictionary size, (b) Time cost involved with keyword dictionary, W = 5000.

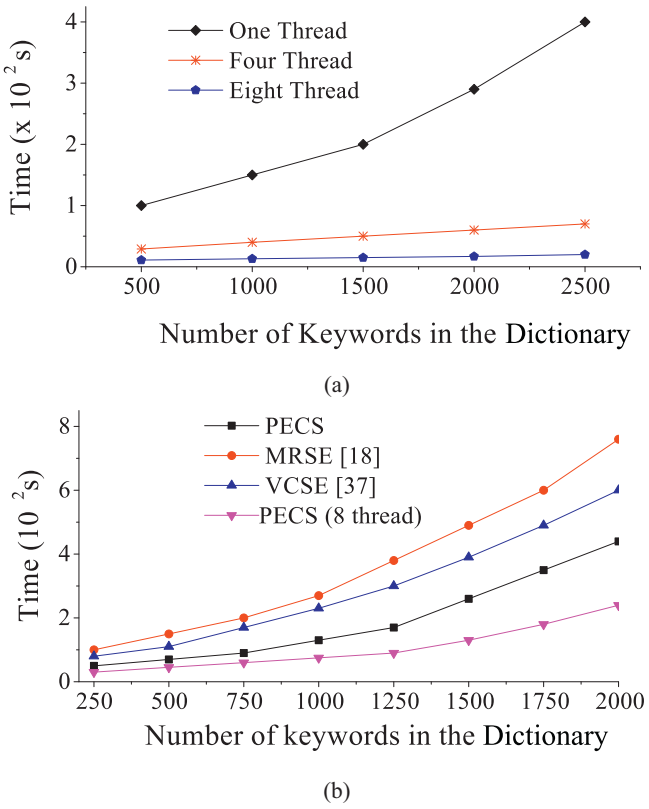

Fig. 7. Query efficiency. (a) Search Efficiency of PECS with Parallel Threads where document size, n = 1000 (b) Time cost of search with document size, n = 2000.

is $O(n)$. The advantage of TPIS is that the membership vector is created based on local keyword dictionary about each partition. As a result, the length of query vector and MV vector are considerably smaller compared to that of global keyword dictionary. For each node, generation of encrypted index vector involves splitting operation and matrix multiplication. However, the dimension of matrices are smaller in TPIS and is $m = \log n$. Therefore, the overall index construction cost is $O(n.m^2)$ and the results are illustrated in Fig. 5.

The time cost for building TPIS index depends on the size of both document collection F, and the keyword dictionary W. Fig. 5(a) and 5(b) shows the time cost for TPIS construction for the environment under test. Results shown in Fig. 5 indicates that the TPIS index construction is almost linear to the document size and is proportional to the keyword dictionary. Moreover, the cost of TPIS construction is compared with that of VCSE [37] and MRSE [18]. VCSE has high index generation cost due to high computational overhead incurred in bilinear pairings. The results in Fig. 5 shows that TPIS has a better time cost of index creation compared to that of VCSE. The better performance is achieved due to the simplicity of MBB node creation and random partitioning of the document collection. The index encryption approach is similar to MRSE. However, PECS shows better time cost compared to that of MRSE due to index organization of TPIS.

6.3. Trapdoor generation

The generation of trapdoor involves splitting of the query vector and matrix multiplication of dimension m and the overall time cost is $O(m^2)$. The advantage of PECS is that due to the random partitioning of large index space into smaller partitions, we get the value of $m = \log n$. Hence, trapdoor generation cost of PECS is better compared to that of VCSE [37] and MRSE [18] as shown in Fig. 6. The number of query keywords have a negligible contribution in the overhead of trapdoor

generation when the dictionary size is fixed as shown in Fig. 6(b) for PECS, whereas it increases almost linearly with the number of keyword in the query for VCSE scheme.

6.4. Query efficiency

6.4.0.4. Query Complexity. The logical AND operator prunes the search domain in a deterministic way which is essential for the retrieval of sensitive data. For a conjunctive query with l terms, $(t_1, t_2, ..., t_l)$, the final result is $query(t_1) \cap query(t_2) \cap ... \cap query(t_n)$. A conjunctive query returns document list containing all query terms. The proposed ranked conjunctive search returns a subset of documents with higher keyword weight matching. The membership vector qualifies for the search algorithm to perform the domain pruning required for AND operator. Hence, tree-based indexing and membership vector helps to filter out search paths to achieve better search efficiency with PECS.

We evaluate the query efficiency of PECS through the experiments and results are shown in Fig. 7. The tree-based index helps to initiate multiple search threads in parallel which adds to the inherent search efficiency of the proposed model. The structure of TPIS permits to have multiple threads for performing a search operation to achieve optimal performance. Assume that there are $m = \log n$ partition and each partition has at most n/m elements. Then the height of the balanced binary tree is $\log m$. Each node requires only membership test before proceeding to lower levels and the time cost is constant. Therefore, the time complexity of search is $O(m.\log m)$. The actual time cost is less than this because all MBB tree nodes may not be searched due to a failure of membership test. Also, initiating a parallel search with the help of multiple threads can increase search efficiency. The search performance is tested with 1, 4, and 8 threads in parallel. Fig. 7(a) shows that with four and eight threads the proposed scheme achieves a considerable improvement in search efficiency. Each partition returns a list of matched documents, and we assume that an aggregate function merges all these lists and returns to the data user. The overhead in merging is negligible since it is implemented as a simple set union.

The search efficiency of PECS is compared to that of VCSE [37], and MRSE [18] which is closely related to ours. The experimental results of query time are shown in Fig. 7(b). Search in PECS can be performed in parallel across different target MBB trees enabling higher search efficiency with four and eight parallel threads. So, the query efficiency of PECS will not significantly drop with the scaling up of documents. VCSE is based on public key cryptosystem and uses bilinear pairing operation which incurs massive computational cost in the process of index creation and search. The index structure of MRSE is not suitable for parallel search, and the performance degrades with the scaling up of documents. As a result, the proposed model ensures improved search efficiency by scaling up of the dataset.

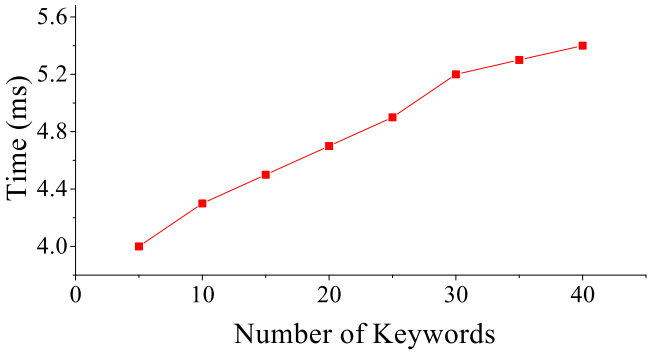

Fig. 8. Update cost of MV vectors in MBB tree, $|F| = 5000$, $|W| = 2000$.

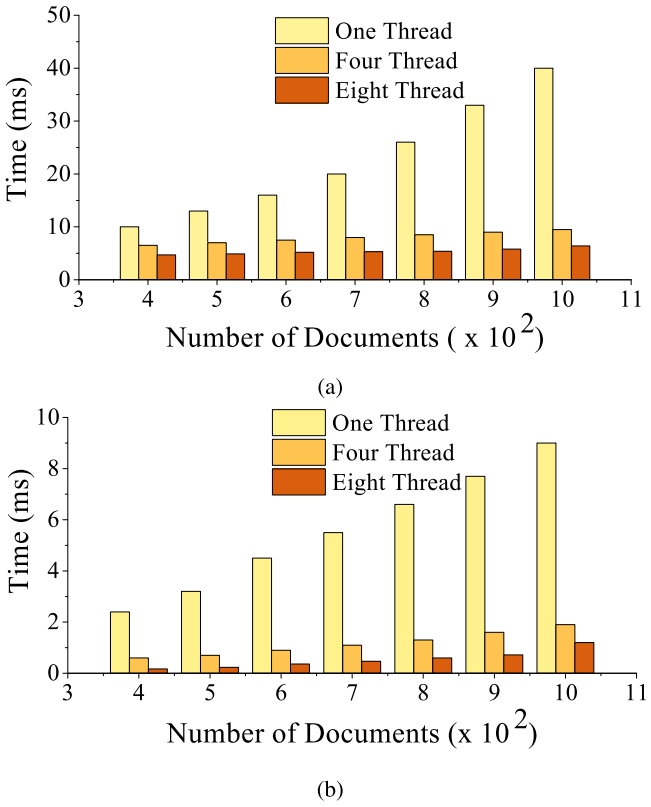

Fig. 9. Time cost of an update operation. (a) Time Cost of Insertion of Documents in PECS, (b) Time Cost of Deletion of Documents in PECS.

6.5. Update efficiency

In the cloud storage, dynamic update of outsourced data is needed for practical importance. We carry out experiments to measure the temporal overhead for inserting and deleting a document. Updating of a leaf node initiates the process to update all $\log \log n$ nodes in the MBB tree. Here, two types of updating can take place. First one is updating the leaf node while keeping the keyword dictionary constant. This process is simple and takes only $O(\log \log n)$ time cost. However, updating with the addition of a new keyword involves more time compared to that of updating as in the first scenario. Here, the keyword is added to the document membership vector and keyword dictionary. Also, add one more MV bit is added to all nodes present in the MBB tree. Hence, the overall cost of doing this is $O(\log n)$. Fig. 8 shows that time cost involved in updating keyword dictionary is proportional to the number of keywords.

PECS is evaluated for the scalability regarding the number of document updating, and the results are illustrated in Fig. 9. Insertion of documents might lead to the addition of keywords in a specific partition, and the nodes of MBB tree are updated. The results in Fig. 9(a) that parallel execution of insertion is more efficient with TPIS which indicates that the proposed indexing structure is scalable towards dynamic insertion of documents. Deletion of nodes is implemented simply. The tree is maintained as a balanced one by adding a dummy leaf node where all MV bits set as 0. Then the change is propagated to all parent nodes of that leaf node. Therefore, deletion involves less time cost compared to that of insertion cost. The time cost for deleting documents is shown in Fig. 9(b). The results show that PECS has minimum temporal overhead for updating which is essential for maintaining a dynamic database.

6.6. Discussion

Data outsourcing demand faster query response and dynamic updating of the database. PECS is an attempt to enhance user search experience by providing a scalable indexing with dynamic update of keyword dictionary. The comprehensive theoretical and experimental evaluation shows the improvement in index cost and search efficiency achieved by the proposed PECS. The major dimensions of improvement obtained are as follows:

- **Index cost:** With document scaling, index creation cost increases more than quadratic for VCSE whereas for both MRSE and PECS it lies between linear and $n\log n$. Also, the index cost of PECS is 10% less compared to that of MRSE due to the support of partitioned index structure.
- **Trapdoor generation:** Due to the local keyword dictionary, PECS achieves the least trapdoor generation cost compared to that of both VCSE and MRSE with keyword scaling. With the increase in the query keywords, both PECS and MRSE have constant trapdoor generation where for PECS it is the least (15ms) compared to MRSE (17ms) for keyword dictionary size, $W = 5000$.
- **Search efficiency:** PECS achieves the best search efficiency compared to that of MRSE and VCSE with the support of parallel search. The gain in the sublinear search is 30% for PECS which is a significant achievement with keyword scaling.
- **Update:** The update cost is less than linear due to the partitioned structure permitting update operation to be executed in parallel across different sub-index and local dictionary.

In practice, PECS handles search in parallel manner to achieve sublinear search efficiency with large volume data. Also it supports document scaling with partition index creation and local keyword dictionary. The reduced sub-index size enables PECS to use matrices of reduced dimension during index creation and trapdoor generation resulting in better time cost. With a huge volume of data, we can reduce the computation cost by parallelizing it across the sub-indexes.

In general, the inclusion of *OR* and *NOT* operators along with *AND* will definitely improve search experience. The ranked conjunctive query will return the subset of documents which contains all query terms. In other words AND operator extend deterministic pruning of the search space. The membership vectors stored at each MBB tree node helps to filter the search path which is essential for faster query response. Disjunctive query expression returns the subset of documents with most matching keywords. But, if we include OR operator then the query expression has to be unfolded into multiple query terms and finally taking the intersection of all result sets. For example, $(q_1 \vee q_2 \vee q_3) \wedge (q_{11} \wedge q_{12} \wedge q_{13})$ results in three matching rules as follows: $q_1 \wedge (q_{11} \wedge q_{12} \wedge q_{13})$, $q_2 \wedge (q_{11} \wedge q_{12} \wedge q_{13})$, and $q_3 \wedge (q_{11} \wedge q_{12} \wedge q_{13})$. Here, membership vectors do not help to filter search path as with AND operator. The result of NOT operator can be represented as $\overline{q_1} \cap \overline{q_2} \cap ... \cap \overline{q_l}$. However, it has got a relevant application with a combination of AND and OR expression. Hence, sophisticated methods are required to prune the search space to provide faster response. The pruning is essential to make sure that only necessary documents satisfying *AND, OR* and *NOT* are retrieved which is considered as a future work.

7. Conclusions and future work

This paper proposes a novel privacy-enhanced search scheme, PECS, which performs the computationally efficient conjunctive search over encrypted data. First, we propose tree based partitioned indexing TPIS to achieve parallel search and dynamic updating with minimum overhead to achieve optimal search efficiency. The indexing structure TPIS enables the CDSP to perform the search in parallel using multiple threads to reduce search time cost. TPIS uses an exclusive membership balanced binary tree to achieve space efficiency in maintaining index vector. The node construction of MBB tree ensures maximum search efficiency. The proposed scheme PECS provides the keyword, query, and index confidentiality and privacy in both threat models. Moreover, the proposed model PECS is compared with other similar models. The experimental analysis shows that PECS obtains better search efficiency and scalability compared to that of other two methods making it suitable for the real environment.

For the future work, we intend to have more boolean operators such as OR and NOT in the query term. PECS could also be modified to have a prioritized search based on users giving different priority levels for the keywords in the query. The personalized keyword priority may help to improve the user search experience. Also, to preserve the privacy of access in the scenario of sensitive data PECS could be modified to hide access pattern. Thus, PECS architecture may be modified to incorporate dynamic allocation of data blocks after each access.

Acknowledgement

This research work is supported in part by the Quality Improvement Programme of All India Council for Technical Education, India. We would like to thank anonymous reviewers and editors for their valuable suggestions that greatly improved the quality of this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.comcom.2018.05.008.

References

[1] V. Chang, Y.-H. Kuo, M. Ramachandran, Cloud computing adoption framework: a security framework for business clouds, Future Gener. Comput. Syst. 57 (2016) 24–41, http://dx.doi.org/10.1016/j.future.2015.09.031.
[2] X. Jiang, X. Ge, J. Yu, F. Kong, X. Cheng, R. Hao, An efficient symmetric searchable encryption scheme for cloud storage, J. Internet Serv. Inf. Secur. (JISIS) 7 (2) (2017) 1–18.
[3] G. Somani, M.S. Gaur, D. Sanghi, M. Conti, R. Buyya, Ddos attacks in cloud computing: issues, taxonomy, and future directions, Comput. Commun. (2017), http://dx.doi.org/10.1016/j.comcom.2017.03.010.
[4] S.K. Pasupuleti, S. Ramalingam, R. Buyya, An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing, J. Network Comput. Appl. 64 (2016) 12–22, http://dx.doi.org/10.1016/j.jnca.2015.11.023.
[5] D. Sánchez, M. Batet, Privacy-preserving data outsourcing in the cloud via semantic data splitting, Comput. Commun. 110 (2017) 187–201, http://dx.doi.org/10.1016/j.comcom.2017.06.012.
[6] W. Song, B. Wang, Q. Wang, Z. Peng, W. Lou, Y. Cui, A privacy-preserved full-text retrieval algorithm over encrypted data for cloud storage applications, J. Parallel Distrib. Comput. 99 (2017) 14–27, http://dx.doi.org/10.1016/j.jpdc.2016.05.017.
[7] N. Kaaniche, M. Laurent, Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms, Comput. Commun. 111 (2017) 120–141, http://dx.doi.org/10.1016/j.comcom.2017.07.006.
[8] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Gener. Comput. Syst. 28 (3) (2012) 583–592, http://dx.doi.org/10.1016/j.future.2010.12.006.
[9] D.X. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, IEEE, 2000, pp. 44–55, http://dx.doi.org/10.1109/SECPRI.2000.848445.
[10] F. Han, J. Qin, J. Hu, Secure searches in the cloud: a survey, Future Gener. Comput. Syst. 62 (2016) 66–75, http://dx.doi.org/10.1016/j.future.2016.01.007.
[11] H.S.G. Pussewalage, V.A. Oleshchuk, Privacy preserving mechanisms for enforcing security and privacy requirements in e-health solutions, Int. J. Inf. Manage. 36 (6) (2016) 1161–1173, http://dx.doi.org/10.1016/j.ijinfomgt.2016.07.006.
[12] H. Yin, Z. Qin, L. Ou, K. Li, A query privacy-enhanced and secure search scheme over encrypted data in cloud computing, J. Comput. Syst. Sci. (2017), http://dx.doi.org/10.1016/j.jcss.2016.12.003.
[13] E.-J. Goh, et al., Secure indexes. IACR Cryptology ePrint Archive 2003 (2003) 216.
[14] Y.-C. Chang, M. Mitzenmacher, Privacy preserving keyword searches on remote encrypted data, ACNS, volume 5, Springer, 2005, pp. 442–455, http://dx.doi.org/10.1007/11496137_30.
[15] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, J. Comput. Secur. 19 (5) (2011) 895–934, http://dx.doi.org/10.3233/JCS-2011-0426.
[16] C. Wang, N. Cao, J. Li, K. Ren, W. Lou, Secure ranked keyword search over encrypted cloud data, Distributed Computing Systems (ICDCS), 2010 IEEE 30th

International Conference on, IEEE, 2010, pp. 253–262, http://dx.doi.org/10.1109/ICDCS.2010.34.
[17] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, H. Li, Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 3025–3035, http://dx.doi.org/10.1109/TPDS.2013.282.
[18] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 25 (1) (2014) 222–233, http://dx.doi.org/10.1109/TPDS.2013.45.
[19] D. Cash, J. Jaeger, S. Jarecki, C.S. Jutla, H. Krawczyk, M.-C. Rosu, M. Steiner, Dynamic searchable encryption in very-large databases: data structures and implementation. NDSS, volume 14, (2014), pp. 23–26.
[20] Z. Xia, X. Wang, X. Sun, Q. Wang, A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2016) 340–352, http://dx.doi.org/10.1109/TPDS.2015.2401003.
[21] C. Orencik, A. Selcuk, E. Savas, M. Kantarcioglu, Multi-keyword search over encrypted data with scoring and search pattern obfuscation, Int. J. Inf. Secur. 15 (3) (2016) 251–269, http://dx.doi.org/10.1007/s10207-015-0294-9.
[22] Z. Fu, K. Ren, J. Shu, X. Sun, F. Huang, Enabling personalized search over encrypted outsourced data with efficiency improvement, IEEE Trans. Parallel Distrib. Syst. 27 (9) (2016) 2546–2559, http://dx.doi.org/10.1109/TPDS.2015.2506573.
[23] X. Ding, P. Liu, H. Jin, Privacy-preserving multi-keyword top-k similarity search over encrypted data, IEEE Trans Dependable Secure Comput (2017), http://dx.doi.org/10.1109/TDSC.2017.2693969.
[24] H. Li, Y. Yang, T.H. Luan, X. Liang, L. Zhou, X.S. Shen, Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data, IEEE Trans Dependable Secure Comput 13 (3) (2016) 312–325, http://dx.doi.org/10.1109/TDSC.2015.2406704.
[25] Z. Fu, X. Wu, Q. Wang, K. Ren, Enabling central keyword-based semantic extension search over encrypted outsourced data, IEEE Trans. Inf. Forensics Secur. 12 (12) (2017) 2986–2997, http://dx.doi.org/10.1109/TIFS.2017.2730365.
[26] Z. Fu, F. Huang, K. Ren, J. Weng, C. Wang, Privacy-preserving smart semantic search based on conceptual graphs over encrypted outsourced data, IEEE Trans. Inf. Forensics Secur. 12 (8) (2017) 1874–1884, http://dx.doi.org/10.1109/TIFS.2017.2692728.
[27] X. Jiang, J. Yu, J. Yan, R. Hao, Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data, Inf. Sci. 403 (2017) 22–41, http://dx.doi.org/10.1016/j.ins.2017.03.037.
[28] Z. Deng, K. Li, K. Li, J. Zhou, A multi-user searchable encryption scheme with keyword authorization in a cloud storage, Future Gener. Comput. Syst. 72 (2017) 208–218, http://dx.doi.org/10.1016/j.future.2016.05.017.
[29] Z. Guo, H. Zhang, C. Sun, Q. Wen, W. Li, Secure multi-keyword ranked search over encrypted cloud data for multiple data owners, J. Syst. Software 137 (2018) 380–395, http://dx.doi.org/10.1016/j.jss.2017.12.008.
[30] Y. Yang, X. Liu, R. Deng, Expressive query over outsourced encrypted data, Inf. Sci. (2018), http://dx.doi.org/10.1016/j.ins.2018.02.017.
[31] D. Boneh, G. Di Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, Eurocrypt, volume 3027, Springer, 2004, pp. 506–522, http://dx.doi.org/10.1007/978-3-540-24676-3_30.
[32] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, ACNS, volume 4, Springer, 2004, pp. 31–45, http://dx.doi.org/10.1007/978-3-540-24852-1_3.
[33] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, Theory of Cryptography Conference, Springer, 2007, pp. 535–554, http://dx.doi.org/10.1007/978-3-540-70936-7_29.
[34] Y.H. Hwang, P.J. Lee, Public key encryption with conjunctive keyword search and its extension to a multi-user system, International Conference on Pairing-Based Cryptography, Springer, 2007, pp. 2–22, http://dx.doi.org/10.1007/978-3-540-73489-5_2.
[35] Y. Miao, J. Ma, X. Liu, Q. Jiang, J. Zhang, L. Shen, Z. Liu, Vcksm: verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings, Pervasive Mob Comput (2017), http://dx.doi.org/10.1016/j.pmcj.2017.06.016.
[36] S. Kamara, T. Moataz, Boolean searchable symmetric encryption with worst-case sub-linear complexity, Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2017, pp. 94–124, http://dx.doi.org/10.1007/978-3-319-56617-7_4.
[37] Y. Miao, J. Ma, F. Wei, Z. Liu, X.A. Wang, C. Lu, Vcse: verifiable conjunctive keywords search over encrypted data without secure-channel, Peer-to-Peer Networking Appl. 10 (4) (2017) 995–1007, http://dx.doi.org/10.1007/s12083-016-0458-z.
[38] Y. Miao, J. Ma, X. Liu, Z. Liu, L. Shen, F. Wei, Vmkdo: verifiable multi-keyword search over encrypted cloud data for dynamic data-owner, Peer-to-Peer Networking Appl. 11 (2) (2018) 287–297, http://dx.doi.org/10.1007/s12083-016-0487-7.
[39] W.K. Wong, D.W.-l. Cheung, B. Kao, N. Mamoulis, Secure knn computation on encrypted databases, Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, ACM, 2009, pp. 139–152, http://dx.doi.org/10.1145/1559845.1559862.
[40] W. Sun, W. Lou, Y.T. Hou, H. Li, Privacy-preserving keyword search over encrypted data in cloud computing, Secure Cloud Computing, Springer, 2014, pp. 189–212, http://dx.doi.org/10.1007/978-1-4614-9278-8_9.
[41] C. Wang, N. Cao, K. Ren, W. Lou, Enabling secure and efficient ranked keyword search over outsourced cloud data, IEEE Trans. Parallel Distrib. Syst. 23 (8) (2012) 1467–1479, http://dx.doi.org/10.1109/TPDS.2011.282.