Indonesian Journal of Electrical Engineering and Computer Science
Vol. 18, No. 1, April 2020, pp. 64~74
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v18.i1.pp64-74

Enhance manet usability for encrypted data retrieval from cloud computing

Fairouz Sher Ali, Hadeel Noori Saad, Falah Hassan Sarhan, Bushra Naaeem
Faculty of Education for Girls, Kufa University, Iraq

Article Info

Article history:

Received Jul 20, 2019
Revised Sep 21, 2019
Accepted Oct 23, 2019

Keywords:

Bloom filter
Inverted index
MANET
Public key encryption
Ranked keyword search
Searchable encryption

ABSTRACT

Cloud computing has become a revolutionary computing model which provides an economical and flexible strategy for resource sharing and data management. Due to privacy concerns, sensitive data has to be encrypted before being uploaded to the cloud servers. Over the last few years, several keyword searchable encryption works have been described in the literature. However, existing works mostly focus on secure searching using keyword and only retrieve Boolean results that are not yet adequate. On the other hand, poor-resources of mobile networks play an important role on all applications area nowadays. Mobile nodes mostly act as information retrieval end which make it important to address this problem. In this paper, we present a secure keyword search scheme based on the Bloom filter (SKS-BF), which enhances the system's usability by allowing ranking based on the relevance score of the search results and retrieves the top most relevant files instead of retrieving all the files. Further, the Bloom filter (BFs) can accelerate a search process involving a large number of keywords. Extensive experiments and network simulation confirm the efficiency of our proposed schemes.

Corresponding Author:

Fairouz Sher Ali,
Faculty of Education for Girls,
Kufa University, Kufa, Iraq.
Email: fairoozm.jaafar@uokufa.edu.iq

1. INTRODUCTION

Many papers are focusing on the security of cloud data storage, which is regarded as an important feature of the quality of service. A currently standard solution to protect data in cloud computing is applying cryptographic techniques to sensitive data. There are very many crypto-graphic primitives that we are able to apply in this manner. In recent years, many attempts have been made to propose efficient and effective schemes to enable searching over encrypted data. A keyword-based search approach has been widely used by traditional search engines, e.g. Google plaintext keyword search. In recent years, secure search over encrypted data has attracted the attention of many scholars. Song *et al* [1] first define and suggest the conception of searchable encryption (SE), which is a cryptographic primitive that allows users to perform a keyword-based search on encrypted files [1-4], as completely as on a plaintext dataset. Searchable encryption can be classified into two fields: symmetric searchable encryption (SSE) [1, 5] and asymmetric searchable encryption (ASE) [6-10].

Modern information retrieval(IR) techniques [11] utilize the ranked-search model to rank all the retrieved documents according to some relevance criteria. Such a model provides precise results by only downloading the top-k relevant files from the whole file collection. Wang *et al.* [12] first discussed the problem of elective yet secure ranked keyword search over encrypted cloud data. This scheme enhances the system's usability by retrieving the matching documents in a ranked order regarding certain relevance criteria

for example keyword frequency. Furthermore, a ranked search can eliminate unnecessary network traffic by receiving only the most relevant data.

As before, searchable encryption is an important solution that allows search directly over encrypted data. Similar to searching over plaintext documents, a common method of searchable encryption schemes is to build a secure index for the whole document collection. Then, in the search phase, which is done on the cloud server side, just the secure index is referenced. Such secure index-based searchable encryption not only enhances the scalability of the searches, but also makes the encryption of each document independent of the searchable encryption scheme used. The inverted index structure is the most common one for the keyword search on encrypted data [3, 13-15].

In terms of Communication cost, Mobile networks (MANET) widespread because of its necessity when network infrastructure are not available or impossible, in recent years mobile nodes are growing rapidly, this lead to the presence limited resource mobile clouds clients. Many attempts have been made to propose efficient and effective schemes to enable searching over encrypted data with mobile-cloud environment [16, 17]. Gupta [18] has stated that MANET may play primary and inevitable role in the growth of cloud-based services. In Ref. [19], the authors express that the attractive feature of cloud based services which depend on eliminate or reduce the requirement of infrastructure-based network, but limited resources of MANET take considerable research area.

Data encryption makes cloud data utilization, e.g. keyword search, a very challenging problem. Retrieving the encrypted dataset first, then searching the decrypted data, is evidently counterproductive and a waste of precious computational and communicational resources. The early encryption schemes didn't concern mobile-clouds (or mobicloud) directly, so researches didn't gave saving energy issue high priority. As mobicloud emerged, an energy consumption is being coming critical, a lightweight algorithm and is suitable in mobile devices. Therefore, Most of the security concerned frameworks offloads cloud servers intensive jobs for saving the limited resources mobile networks [20]. The search process via keyword is one of searching strategies which completed at the cloud side and on the encrypted data reducing the communication overheads

As mentioned earlier, the first public key scheme for keyword search over encrypted data (PEKS) is presented by Boneh *et al.* [6]. The scheme does not handle the issue of secure ranked keyword search over encrypted cloud data [21]. To solve this issue and enable cloud servers to perform secure search without knowing the original value of either the keywords or trapdoors, we present a secure ranked keyword search scheme based on the Bloom filter (SKS-BF).

The contributions of this paper can be summarized as follows:

a) Building a secure inverted index based on Bloom filter [22] and public key searchable encryption scheme.
b) Extending the secure searchable index to a rank-oriented searchable index. To do so, we assign numerical relevancy scores to each document matching a given trapdoor. We build the secure keyword search technique based on the bilinear map, which allows data users to protect the privacy of the relevance scores and of the trapdoor, and show that the proposed scheme is secure in the random oracle model (Rom) under the Bilinear Inverse Diffie-Hellman Problem BIDHP.
c) Enhancing the usability of MANET through the reduction of data communication over-head in information retrieval process.

2. LITERATURE REVIEW

The first attempt of Private Information Retrieval (PIR) was made by Chor *et al* [23]. In recent times, Groth *et al.* [24] introduced a multi-query PIR technique with constant communication rate.

Classical searchable encryption has been widely elaborated as a cryptographic primitive, with a concentration on security definition formalizations and efficiency enhancements. The method of searchable encryption was first proposed by Song *et al.* [1]. The suggested method supports single keyword search without index, which means the server must scan the entire file to find the search result. Goh *et al* [2] proposed constructing a keyword index for each file and using Bloom filters to speed up the search. Curtmola *et al*. [3] suggested the first inverted index based encrypted searchable index. Their scheme includes constructing indices for each trapdoor of a keyword, and using hash tables as an alternative technique to searchable encryption. The file list for each keyword is encrypted and inserted into an array. Based on [3], Kamara *et al.* [13] proposed the concept of dynamic searchable encryption and built an encrypted inverted index that handles dynamic operations such as document updates, but their scheme is very complex to apply. Therefore, as an improvement, Kamara *et al*. [25] presented a novel search technique based on a tree-based index. This technique can treat dynamic update on document data buffered in leaf nodes. Searchable encryption has also been regarded in the public-key setting. The first public key scheme for keyword search

over encrypted data was suggested by Boneh *et al*. in [6]. In this scheme, any user with the public key can write to the data stored on the cloud server, but only trusted users with the secret key can search.

The notion of secure ranked search over encrypted data was first proposed by Wang *et al.* [12]. [12, 26] allowed data users to search only with single keyword query and retrieve the top-*k* relevant documents. Cao *et al.* [27, 28] introduced the first privacy-preserving multi-keyword ranked search scheme, in which documents and queries are formed as vectors of dictionary size. With coordinate matching, the documents are ranked according to the number of matched query keywords. These techniques do not take the significance of the different keywords into consideration. Therefore, they are not precise enough. Orencik *et al*. [29] proposed a secure multi-keyword search scheme employing local sensitive hash (LSH) functions to collect similar documents. The LSH protocol is convenient for similar search, but cannot give an accurate ranking. Zhang *et al*. [30] introduced a method to deal with secure multi-keyword ranked search in a multi-user form. In this method, various data users use different secret keys to encrypt their keywords and documents while authenticated data users can query without knowing the keys of these various data users. The authors proposed an Additive Order Preserving Function to return the most relevant search results. Wang *et al.* [31] introduced a public-key searchable encryption method based on an inverted index. Their method preserves the high search efficiency inherited from the inverted index, while removing the one-time-only search limitation of the previous techniques. Recently, Chen *et al.* [32] presented a hierarchical clustering technique to support most search semantics and also to meet the demand for fast ciphertext search within a big dataset. Their technique clusters the documents based on the minimum relevance threshold, and then splits the resulting clusters into sub-clusters until the constraint on the maximum cluster size is reached.

The methodology of optimized routing for cloud-networks using ZRP (Zone Rout Protocol) aims at minimizing routing overheads and prevents redundancy on all nodes in zone by adopt-ing shortest distance between the nodes as the elected best path and discard other paths from the routing table[33-35]. Lee *et al*.[36] proposed a combined compression technique. It incorporates routing process with compressive sensing for energy aware data gathering in wire-less sensor networks. The Simulation results exhibit the electiveness of the proposed combined technique which outperform the standard compressed sensing. Many compression techniques had been invented to solve the energy limitation of mobile nodes, they vary upon compression level from link, data, and protocol's headers [37-41].

3. PRELIMINARY

3.1. Inverted Index

Most current IR techniques [11] utilize an index of the document collection to speed up the search. The inverted index (also referred to as the postings document) is one of the most popular data structures used in information retrieval systems [42]. An inverted index includes multiple inverted lists. One inverted list stores a list of mappings from a keyword w to its corresponding set of document IDs that include this keyword. Additional information can be inserted in the inverted list, like the numerical statistics of the keyword (to support result ranking), each document in the inverted lists is assigned a numerical relevance score. Such scores are used to decide which document is more relevant to the corresponding keyword. The *TFXIDF* method is widely used to calculate the relevance scores. For the given word *w* and the document *D*, TF (term frequency) indicates the number of times in which *w* appears in the document *D*, while IDF (inverse document frequency) refers to the ratio of the whole number of documents in the collection to the number of documents that contain the keyword *w*. Among several *TFXIDF* formulas, we choose the following formula to calculate the relevance of the Document F_d with regards to the keyword w:

$$Score(w, Fd) = \frac{1}{|F_d|} . (1 + lnf_{d,w}) \quad (1)$$

where *w* indicates the searched keyword, $f_{d,w}$ denotes the term frequency *TF* of keyword *w* in document F_d, and $|F_d|$ is the length of the document F_d, obtained by counting the number of indexed terms, functioning as the normalization factor.

The essential advantage of using an inverted index comes from its search efficiency, especially for a large volume of documents. The search process is performed on a much smaller document set, which comprises the inverted lists that correspond to the query keywords.

3.2. Complexity Assumptions

The symbol *G* is used to denote a group which is a set with some binary operation. Definition 3.1. The number of items in G is called the order of G. A group G is finite if the number of items is finite.

Definition 3.2. A group G is cyclic if there is an item v ∈ G such that for each x ∈ G there is an integer i with $x = v^i$. Such an element v is called a generator of G.

The main groups used in this paper are Z_n, G_x and G_y. Z_n denotes the set of integers under addition modulo *n*, G_x is an additive group and G_y is the related multiplicative group. Both G_x and G_y are cyclic groups of large prime order *r* related to elliptic curves over finite fields.

We now review the bilinear pairing that plays a crucial role in building the efficient PEKS scheme, introduced in [6]. Definition 3.3. (Bilinear Pairing [43]) Let G_x and G_y be two cyclic groups with a prime order r, v is a generator of G_x and G_y. Then we say $\hat{e}: G_x \times G_x \rightarrow G_y$ is a bilinear map if the following properties are satisfied:

a) Bilinearity: $\hat{e}(v^a, v^b) = \hat{e}(v, v)^{ab}$ for all $a, b \in Z_n$.
b) Non-degenerate: $\hat{e}(v, v) \neq 1$.
c) Computable: given $v, h \in G_x$ there is a polynomial time algorithm to compute $\hat{e}(v, h) \in G_y$.

Below, we review the definition of the Bilinear Diffie–Hellman (BDH) problem associated with the bilinear pairings [43].

Definition 3.4. (Bilinear Diffie-Hellman (BDH) Problem) Let G_x, G_y be two groups of prime order r, v be a generator of G_x, $\hat{e}: G_x \times G_x \rightarrow G_y$ be an admissible bilinear map and AR be an attacker algorithm. The BDH problem in (G_x, G_y, $\hat{e}$) is as follows: Given (v, v^a, v^b, v^c) for some a, b, c ∈ Z_n, compute $\hat{e}(v, v)^{abc} \in G_y$. An algorithm AR has advantage in solving BDH in G_y if $Pr[AR(v, v^a, v^b, v^c) = \hat{e}(v, v)^{abc}] \geq \epsilon$.

4. PROBLEM FORMULATION

4.1. System and Threat Model

Our scheme includes three different entities, a data owner *DW*, a cloud server *CS*, and data users *DU*. We suppose the authorization between the data owner and data users is appropriately done. *DW* has a collection of n data files $FC = (f_1, f_2, \ldots, f_M)$ that he wants to outsource on CS in encrypted form. To do that, before outsourcing, *DW* will first build a Secure Searchable Index SSI from a set of *m* different keywords $W = (w_1, w_2, \ldots, w_N)$ extracted from the file collection *FC*, and store both the index SSI and the encrypted file collection ENC_{FC} which are encrypted using a standard symmetric algorithm like *AES* [44] on the cloud server.

DU is authorized to access the files of *DW*, he generates a search request (trapdoor *Tw*) of the (keyword *w*) and submits it to the *CS*. Upon receiving the trapdoor *Tw* from the data user, the server searches over the index SSI, and retrieves the corresponding set of ranked encrypted files. After that, *DU* can decrypt the files using the shared secret key.

To minimize the communication cost, the data user may send an optional value k along with the trapdoor *Tw* so that the cloud server only sends back the top-*k* documents that are most relevant to the search request. Additionally, upon receiving the updated information from *DW*, the cloud server needs to update the searchable index SSI and file collection *FC* according to the received information.

In our scheme, we treat the cloud server as an honest-but-curious adversary. In other words, the server follows our proposed algorithm faithfully, but it is anxious to know the keywords, the contents of the encrypted files, and the relevance scores. We suppose that the authorized data users are fully trusted by the data owner and they are authorized to access all the files through search operations.

5. CONSTRUCTION OF THE SKS-BF SCHEME

5.1. Notations

a) FC: the file collection to be uploaded, expressed as a set of M data files $FC = \{f_1, f_2, \ldots, f_M\}$.
b) W: the different keywords extracted from the file collection FC, expressed as a set of N keywords $W = \{w_1, w_2, \ldots, w_N\}$.
c) Enc_{FC} : the encrypted file collection for FC, expressed as $Enc_{FC} = \{Enc_{f_1}, Enc_{f_2}, \ldots, Enc_{f_M}\}$.
d) FC_{wi}: the identifier set of M files in Enc_{FC} that contain keyword w_i and can help uniquely locate the actual file, expressed as $FC_{wi} = \{ID_{w_i f_1}, ID_{w_i f_2}, \ldots, ID_{w_i f_M}\}$.
e) Enc_F: the collection of k retrieved documents from the cloud server contained the key-word, denoted as $Enc_F = \{Enc_{f_1}, Enc_{f_2}, \ldots, Enc_{f_k}\}$.
f) IL_{w_i}: the inverted list which stores a list of mappings from keyword w_i to its corresponding set of document IDs that contain this keyword, and their scores $Sc_{w_i f_j}$, $IL_{w_i} = \{(ID_{w_i f_1} \| Sc_{w_i f_1}), (ID_{w_i f_2} \| Sc_{w_i f_2}), \ldots (ID_{w_i f_M} \| Sc_{w_i f_M})\}$
g) InvLis: the index constructed from the file collection, including a set of inverted lists IL_{w_i}, expressed as $InvLis = \{IL_{w_1}, IL_{w_2}, \ldots, IL_{w_N}\}$.

h) Qw: a subset of W, representing the keywords in a search request, denoted as Qw={qw_1, qw_2, …, qw_l}.
i) Tw: the trapdoor set generated by the data user as a search request of query set Qw, expressed as Tw = {tw_1, tw_2, …, tw_l}.

5.2. Details of the Scheme

In the present paper, we build an inverted index from the file collection *FC*, to speed up the search process. Also, we employ the Bloom filter to accelerate the search process over this index. On the other side, to simplify the ranking task, we need to upgrade the searchable index SSI to a rank-oriented searchable index. To do that, we attach relevance scores with each keyword-file entry in the searchable index.

Unlike the previous techniques, the data owner ranks the matched documents according to relevance scores, since directly uploading relevance scores will leak lots of significant frequency information against the keyword privacy. Hence, the essential issue here is how we can encrypt these numerical scores while preserving their ability to rank the relevant documents. In our scheme, we closely follow the PEKS algorithm to encrypt the inverted index to avoid leaking the relative order of the relevance scores to the cloud server, except for a slight modification in the structure of PEKS, which involves using the Bloom filter. Additionally, in this scheme, *DU* can implement top-k retrieval with one round trip for each search request without waiting for the time of two round trips for each search request, this will avoid unnecessary communication cost. Also observe that in this way, *CS* still knows nothing about the values of the numerical scores, but it learns that the required documents are more relevant than the ones not required.

Our scheme consists of four algorithms: a key generation algorithm *Key-Gen*, file collection encryption algorithm *Enc-FC*, an index construction algorithm *SSI-Const*, a trapdoor generation algorithm *Tw-Gen*, and a searchable index algorithm *SSI-Sear* which are scattered among three phases, the data owner phase, the data user phase and the cloud server phase.

5.2.1. Data Owner Phase

This phase includes three algorithms as detailed below:

a) Key-Gen(θ): DW initiates the scheme by using a key generation algorithm Key-Gen which takes the security parameter θ which determines the size of G_x and G_y, as input to create the following public parameters PP: r as a large prime number, two groups G_x and G_y of order r, V is a random generator of G_x, a bilinear map $\hat{e}: G_x \times G_x \rightarrow G_y$, three cryptographic hash functions H_1, H_2, H_{bloom}, two public keys, one for data owner DW_{PK} and other for data user DU_{PK} , and two secret keys SK: α and β.
b) Enc-FC(FC): To protect the file collection privacy, FC should be encrypted before uploading them onto cloud server which is not within their trusted domains. To do that, the data owner executes a file collection encryption algorithm Enc-FC to encrypt each file f∈FC using AES algorithm [44]. Each file f_i comprising of a unique identifier $ID_i \in \{0, 1\}^n$.
c) SSI-Const (PP, FC): The data owner DW runs an index construction algorithm SSI-Const to build a Secure Searchable Index SSI for all searchable keywords W, which are stored at the service provider, which will help DW to perform a keyword search by calling this algorithm. Firstly, the algorithm scans the file collection FC and extracts the distinct keywords W from FC. Then the algorithm constructs the identifier collection of M files in Enc_{FC} that contain the keyword w_i as $FC_{wi}=\{ID_{w_i f_1}, ID_{w_i f_2}, \ldots, ID_{w_i f_M}\}$. For each file in FC_{wi}, SSI-Const algorithm computes a numerical relevance score Sc according to (1). The computed scores are used to determine which file is more relevant to the corresponding keyword. Next, it attaches the relevance scores with the corresponding files as $(ID| Sc_{w_i f_j})$ and stores them in the Inverted List IL_{w_i}. Then, the algorithm sorts each Inverted List $IL_{w_i} \in$InvLis according to the scores of all the files containing w_i.

To secure the relevance scores and the keyword, we apply bilinear maps on elliptic curves [45]: we use two groups G_x and G_y of prime order r and a bilinear map $\hat{e}: G_x \times G_x \rightarrow G_y$. We also need two hash functions $H_1 : \{0, 1\}^* \rightarrow G_x$ and $H_2 : G_y \rightarrow \{0, 1\}^{logr}$. We use this bilinear map with each relevance scores and keyword w_i as $Enc\text{-}Sc_{wifj} = H_2(\hat{e}\ (DU_{PK}, H_1(Sc_{wifj}^{\alpha}))$ and $Enc\text{-}w_i = \alpha H_1(w_i)V + \alpha(DU_{PK})$ respectively. Then, *DW* stores the Inverted Lists *InvLis* in SSI. Finally, the data owner sends the SSI and the encrypted documents set Enc_{FC} to the cloud server.

On the other hand, to secure the keyword w_i in each Inverted List IL_{wi}, *DW* creates one Bloom filter *BF* for all keywords: this filter consists of an array of *x* bits, and uses *p* independent hash functions h_1, …, h_p. The filter allows the data owner to perform key-word searches efficiently, but could result in some false positive retrievals. A classical Bloom filter may reveal information about the keyword, since the hash functions are publicly known. So, a suitable solution to create a searchable index using the Bloom filter is to instead index each keyword by its encrypted image. Algorithms 1, 2 and 3 below show the key generation algorithm, file collection encryption algorithm and the index construction algorithm.

Algorithm 1-Key-Gen(θ)
1: generate a prime r, and select a random generator V of G_x,
2: choose two cyclic groups $(G_x,+)$, $(G_y,.)$ of order r, and construct a bilinear map $\hat{e}: G_x \times G_x \rightarrow G_y$,
3: specify two hash functions $H_1 : \{0,1\}^* \rightarrow G_x$, $H_2 : G_y \rightarrow \{0,1\}^{\log r}$, where H_1 and H_2 are random oracles, and select p hash functions for the bloom filter $H_{bloom}=\{h_1,\ldots, h_p\}$,
4: pick a random $\beta \in Z_r^*$ as a DU private key, calculate the corresponding public key $DU_{PK} = \beta V$,
5: pick a random $\alpha \in Z_r^*$ as a DW private key, calculate the corresponding public key $DO_{PK} = \alpha V$,
6: return a public parameter $PP = \{G_x, G_y, r, V, H_1, H_2, H_{bloom}, \hat{e}\}$, and a secret keys $SK = \{\alpha,\beta\}$.

Algorithm 2-Enc-FC(FC)
1: for each file $f_i \in FC$ do
- encrypt f_i using (AES) algorithm.

Algorithm 3-SSI-Const(PP, FC)
1: Initialization:
- scan FC and extract the distinct keywords $W = \{w_1, w_2, \ldots, w_N\}$ from file collection FC,
2: Construct inverted lists:
- for each $w_i \in W$, construct FC_{wi} where $i \in [1, N]$,
* for each $ID_{wif\,j} \in FC_{wi}$ where $j \in [1, M]$,
- compute the score for file $ID_{w_i f_j}$ according to (1), denoted as $Sc_{w_i f_j}$,
- store the $ID_{w_i f_j}$'s identifier with score $Sc_{w_i f_j}$, $(ID| Sc_{w_i f_j})$ in the Inverted List IL_{wi},
* sort the inverted list IL_{wi} according to scores of all files contained w_i,
* encrypt $Sc_{w_i f_j}$ as $Enc\text{-}Sc_{w_i f_j} = H_2(\hat{e}(DU_{PK}, H_1(Sc^{\alpha}_{wifj})))$,
3: Secure the keyword in each inverted list:
- for each $IL_{wi} \in InvLis$,
* encrypt w_i, set the encrypted keyword as $Enc\text{-}w_i = \alpha H_1(w_i)V + \alpha(DU_{PK})$,
* compute $R_1 = H_2(\hat{e}(V, V)^{\alpha})$,
4: store the inverted lists InvLis and R_1 in SSI
5: send the encrypted files Enc-FC and the index SSI to the server.

5.2.2. Data User Phase

This phase includes one algorithm as detailed below:

Tw-Gen(PP, Qw): DU will run a trapdoor generation algorithm Tw-Gen to retrieve only the encrypted files containing keywords, DU will get the public key DO_{PK} and create a set of trapdoors $Tw = \{tw_1, tw_2,\ldots, tw_l\}$ from a set of keywords $Qw = \{qw_1, qw_2,\ldots, qw_l\}$. DU will compute the trapdoor of the search request of Qw using his secret key, $tw_i = (H_1(qw_i)+\beta)^{-1}V$ [45]. Finally, data user submits Tw to the cloud server. Algorithms 4 below show the trapdoor generator algorithm.

Algorithm 4- Tw-Gen(PP, Qw)
1: for each $qw_i \in Qw$ where $i \in [1, l]$,
2: compute trapdoor using DU's private key as: $tw_i = (H_1(qw_i) +\beta)^{-1}V$,
3: send the generated trapdoors Tw and the corresponding top-k to the server.

5.2.3. Cloud Server Phase

This phase includes one algorithm as detailed below:

SSI-Sear (SSI, *Tw*, *k*, *PP*): upon receiving the trapdoor *Tw*, the server will call the a searchable index algorithm *SSI-Sear* on the searchable index *SSI* and return the associated Bloom filter *BF*, then compute independent hash functions $H_d(R_1)$ where $d = 1\ldots p$ and $R_1 = H_2(\hat{e}(V,V)^{\alpha})$, and compute $H_d(R_2)$ where $R_2 = H_2(\hat{e}(tw_j, Enc\text{-}w_i))$. Then server tests *BF* in all *p* locations. If all *p* locations of all independent hash functions in *BF* are 1, the server returns the relevant encrypted files corresponding to tw_j to *DU*.

Once *DU* receives the encrypted files from *CS*, he/she decrypts each retrieved document $Enc_{fi} \in Enc_{FC}$ using the *AES* encryption method algorithm. Algorithm 5 below show the search index algorithm.

Algorithm 5- SSI-Sear(SSI, Tw, k, PP)
1: compute the length x of the Bloom filter BF,
2: create the Bloom filter BF of x zero-bits,
3: for each $tw_j \in Tw$ where $j \in [1, l]$,
4: for each $Enc\text{-}w_i \in SSI$ where $i \in [1, N]$,

5: compute $R_2 = H_2(\hat{e}(tw_j, Enc\text{-}w_i))$,
6: compute independent hash functions: $bf_d = H_d(R_1)$ for $d \in [1, p]$,
7: set $BF[bf_d]=1$,
8: compute independent hash functions: $H_d(R_2)$ for $d \in [1, p]$,
9: if all p locations of all independent hash functions in Bloom filter BF are 1, CS gets the ranked search results, then sends the ranked ID list of top-k encrypted documents Enc_F to the data user.

6. SECURITY ANALYSIS

Here, we analyse the security characteristics using the scheme we described above. In terms of privacy of the keywords and relevance scores, we construct the scheme based on bilinear pairing to protect the privacy of the trapdoors and relevance scores. Below, we prove the security of the keyword encryption process. Theorem 6.1. The proposed SKS-BF scheme is semantically secure against chosen-keyword attacks under the Bilinear Inverse Diffie-Hellman Problem (BIDHP).

Proof**.** Let AR be an attacker that wins the security game (breaking the SKS-BF scheme) with an $\in$ advantage. Suppose AR makes Qr_{H1} and Qr_{H2} hash function queries and Qr_T trapdoor queries. Also, we suppose there is an algorithm *CH* that breaks the BIDH problem with advantage at least $\grave{\in}$, and propose CH is given (r, G_x, G_y, $\hat{e}$,V,V). Then CH's goal is to solve a BIDHP. CH interacts with the forger AR in the security game as follows:

At any time, algorithm AR queries the random oracle H_1 or H_2.

1) H_1-queries: To respond to this type of query, the challenger CH maintains H_1, a list$<W_j,\lambda_j, E_j>$. The list is initially empty.

   When AR queries Wi from the random oracle H1, and algorithm CH responds as follows:

   a) If Wi already appears in the list H1, then CH responds with λi. Otherwise, CH generates a random coin $Ei \in \{0,1\}$. If Ei = 0, then CH computes λi= bV for a randomly selected $b \in Z_r^*$, otherwise, Ei = 1, CH selects a random element a and computes λi = a-1V.

   b) In both cases, the challenger adds the tuple < Wi, λi , Ei > to the list H1 and responds with H1(Wi) = λi.

   c) When AR requests an encryption of keyword W, Algorithm CH calls the above algorithm to respond to H1-queries to get $\lambda i \in Gx$. Then he searches H1 for the keyword Wi. If Ei = 1, then CH aborts. Otherwise, H1(Wi) = bV, and then CH computes

   Enc^*_{Wb}= H1(Wi)X +α DUPK
   α (bV)V +α DUPK
   α (bV)V + αβ V
   α V(bV+β)

   d) CH can build a searchable index SSI by executing the algorithm SSI-Const(PP, FC). Then it returns the index SSI to AR.

2) - H_2-queries: The challenger maintains H_2, a list (γ,φ). The list is initially empty. At any time, AR issues a query γ to H_2. CH checks whether γ appears in H_2 in a pair (γ,φ). If not, the challenger CH picks a random value φ, adds the pair (γ,φ) to H_2, and answers AR with $H_2(\gamma) = \varphi$.

3) Trapdoor queries: When AR makes a query for the trapdoor of the keyword W_i, CH runs the above algorithm to respond to H_1-queries to get $\lambda_i \in G_x$, then searches the H_1-list for the query. If $E_i = 1$, then CH aborts. Otherwise, $H_1(W_i) = bV$, and then CH computes

   Tw = (H1(Wi) +β)-1V
   (bV +β)-1V,

   where CH does know β, and answers AR with Tw.

4) Challenge: Algorithm AR selects and sends a pair of keywords W0 and W1 to CH on which it wishes to be challenged, and AR must not have asked previously for the trapdoors of any of the words W0 or W1. After receiving (W0, W1) from the attacker, CH calls the above algorithm twice to respond to H1-queries to get (W0, λ0, E0) and (W1, λ1, E1). If both E0 and E1 are 0, then CH reports failure and terminates. Otherwise, CH randomly picks $ð \in \{0,1\}$ such that Eð = 1, and then CH responds with the challenge ciphertext Encw and R1 where $R1 \in \{0,1\}$logr C = (Encw, R1). The decryption of the ciphertext is

$Dec_{ð}^{*}$= H2(ê($Tw_{ð}^{*}$,$Enc_{W_{ð}^{*}}$))
= H2(ê(bV, a-1V))
= H2(ê(V, V)$^{a^{-1}b}$)

by the definition $Dec_{ð}^{*} = R_1^*$. The challenger creates the secure index SSIð by running the algorithm SSI Const(PP, FC) and sends SSIð as a challenge to AR.

5) More queries: After the above challenge query, AR can do additional trapdoor queries, with same restriction that $W_i \neq W_0$, W_1, CH answers these queries as before.
6) Output: Eventually, AR outputs its guess ð́ ∈ {0, 1} for ð.

7. PERFORMANCE OF OVERALL SKS-BF SYSTEM

In this section, we measure the efficiency of the proposed scheme with MANET resources usability in terms of retrieved data and corresponding energy saving. Also, we evaluate our scheme and compare its performance with PPSE composed in [46]. The entire experiment design is implemented using MATLAB application and NS2 2.3 simulator on a Windows 8 server with an Intel(R) Core(TM) i5-3032M CPU at 2.60GHz. The bloom filter and encrypted data retrieval was performed with MATLAB, and the entire network design and residual energy calculation were performed with ns2 simulation.

7.1. Evaluation Results

To build a secure index with ranked keyword search, an ordinary posting list attaches a relevance score to each posting entry. Our method replaces the original values of scores with the encrypted ones. The size of the posting lists is absolutely linear in the size of the keyword dictionary. The additional bits of encrypted relevance scores are not an important issue because of the cheap storage overhead on cloud servers. On the other hand, our scheme takes a long time to build the secure index; this is because the security of SKS-BF heavily relies on the bilinear pairing operation and operations over elliptical curves. So, we do not discuss the index construction time here, we suppose the data owner has built the secure searchable index already. We focus on the efficiency of the search: after receiving the trapdoor from the user, the server needs to compare the trapdoor with the searchable index. The consumed time is shown in the following figures.

As the encrypted scores are ranked by *DW*, the cloud server can process the top-*k* retrieval almost as fast as in the plain text domain. Figure 1 shows our search time overhead against increasing values of *k*, for the same index with ten queried keywords for different numbers of retrieved files with the same file collection M = 10000, and dictionary keyword N = 4000.

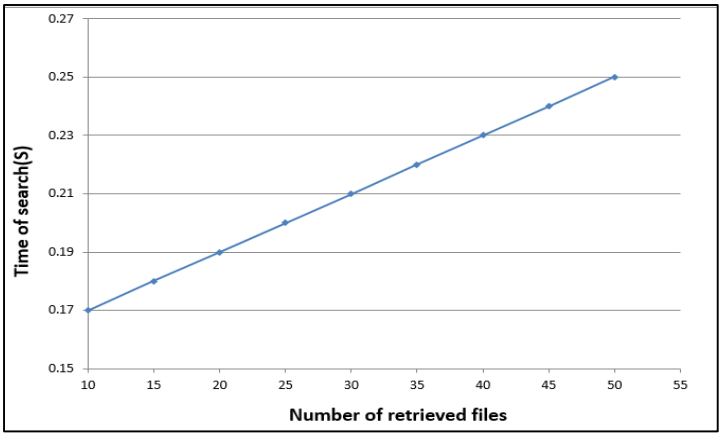

Figure 1. The time overhead of query when l = 10; M = 10000 and N = 4000

In terms of search efficiency evaluation, we compare the search time complexity with the PPSE scheme [46]. The main computation to perform a query in the cloud server consists of computing and ranking the similarity scores for all documents in the dataset and choosing the top-*k* results from all the scored documents. For SKS-BF, query execution in the cloud server includes one protocol (Locate Keywords and returns the ranked *ID* list of top-*k* encrypted files Enc_F to *DU*).

Figure 2 shows that, given the same number of queried keywords (l = 10) and the same size of data files (M = 10000), when the size of keyword dictionary changes from 4000 to 12000, the time overhead of the search also increases linearly for the scheme PPSE, while it remains constant for SKS-BF.

Figure 3 shows that, given the same number of queried keywords (l = 10) and the same size of the keyword dictionary (N = 4000), with different numbers of documents, the time overhead of the search increases linearly for PPSE, while it remains constant for SKS-BF. Furthermore, the search time of SKS-BF is sensitive to the size of the keyword dictionary, and it is insensitive to the size of the dataset, while PPSE suffer a quadratic growth as the size of the dataset increases and the size of the keyword dictionary increases. However, from the discussions above, we can put the conclusions that our technique has less search time consumption than PPSE.

In terms of communication cost which is the core problem of MANET, it is obvious that the proposed method reduces the retrieved files, and accordingly the communication processes saves the node's energy as shown in Figure 4. The energy overhead decreases linearly. The proposed scheme avoids unnecessary communications, and therefore has highly contributed in energy saving.

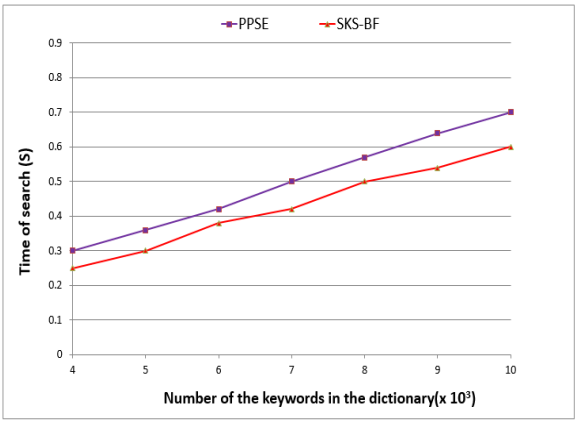

Figure 2. The time overhead of query when l = 10; M = 10000 and k = 50

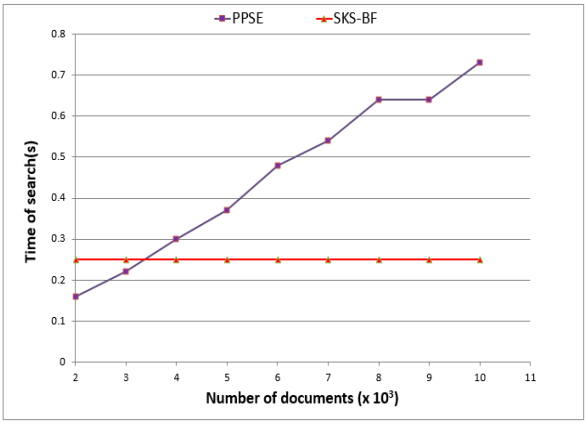

Figure 3. The time overhead of query when l = 10; N = 4000 and k = 50

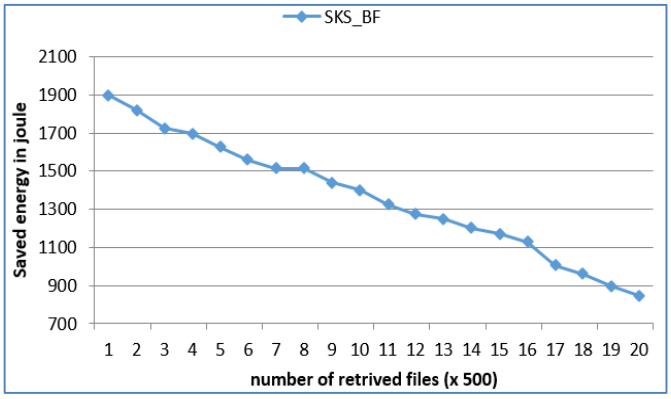

Figure 4. The saved energy vs number of retrieved files

The simulation environment, as shown in Table 1, comprise 50 nodes with initial battery power 2000 joule. All the simulation's traces take place at the sending node. SKS-BF takes place in 20 simulation sessions on NS2 for multiples of 500 retrieved files for each search request. Downloading the top-*k* relevant files from the whole file collection is an important concentration point from MANET developer point view.

Table 1. Simulation Environment

No.nodes	50
Data stream	CBR
Battery initial energy	2000 joule
Routing protocol	DSDV
Packet size	512 byte

8. CONCLUSION

In this paper, we suggest a novel construction of a public key searchable encryption method. We use the inverted index and Bloom filter to speed up the search processes with a large amount of keywords. The scheme is efficient because it relies on pairing operation on elliptical curve. Because the search pattern has been identified as an important security threat in searchable encryption techniques, the proposed scheme features a probabilistic trapdoor construction algorithm to protect the search pattern. Finally, the experimental and simulation results demonstrate that the proposed scheme not only properly tackles the secure ranked keyword search issue, but also brings an enhancement in the reduction of data communication through the search and retrieval processes.

REFERENCES

[1] DX. Song, *et al*., "*Practical techniques for searches on encrypted data*," In Proceedings of IEEE Symposium on Security and Privacy. IEEE, Berkeley, California, 2000, pp.44-55.

[2] E-J. Goh, "Secure indexes," *Cryptology ePrint Archive, Report* 2003/216, 2003.

[3] R. Curtmola, *et al*., "*Searchable symmetric encryption: improved definitions and efficient constructions*," In: Proceedings of the 13th ACM conference on Computer and communications security. ACM, Alexandria, VA, USA, 2006, pp. 79-88.

[4] F. SherAli and S. F. Lu, "*Searchable Encryption with Conjunctive Field Free Keyword Search Scheme*," International Conference on Network and Information Systems for Computers (ICNISC), Wuhan, 2016, pp. 260-264.

[5] F. SherAli and S. F. Lu, "Symmetric Key Encryption with Conjunctive Field Free Keyword Search Scheme," *British Journal of Mathematics & Computer Science*, Vol.16, No.6, pp.1-11, 2016.

[6] D. Boneh, *et al.*, "*Public key encryption with keyword search*," in Proc. of EUROCRYP04, Vol.3027 of LNCS. Springer, 2004.

[7] A.V. Vora, and S. Hegde, "Keyword-based private searching on cloud data along with keyword association and dissociation using cuckoo filter," *International Journal of Information Security; Springer, Berlin/Heidelberg*, 2018, pp.1-15.

[8] S. Yin, L. Teng, J. Liu, "Distributed Searchable Asymmetric Encryption", *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, Vol.4, No.3, pp.684-694, December 2016.

[9] F. SherAli and S. F. Lu, "Public Key Encryption with Conjunctive Field Free Keyword Search Scheme," *International journal of computers & technology*, Vol.15, No.14, pp.7423-7434, 2016.

[10] F. SherAli, "Public Key Encryption with 'Fixed and Short Length' Keyword Search," *International Journal of Computer Applications*, Vol.154, No.4, pp.1-8, 2016.

[11] C. D. Manning, *et al*., "Introduction to Information Retrieval," *Reading, MA: Cambridge UP*, 2008.

[12] C. Wang, *et al*., "*Secure ranked keyword search over encrypted cloud data*," in Proc. IEEE Distributed Computing Systems (ICDCS10), Genoa, Italy, 2010, pp. 253-262.

[13] S. Kamara, *et al*., "*Dynamic searchable symmetric encryption*," in Proceedings of the 2012 ACM Conference on Computer and Communications Security, ser. CCS 12. New York,NY, USA: ACM, 2012, pp.965-976.

[14] M. Naveed, *et al*., "*Dynamic searchable encryption via blind storage*," in Proceedings of the 2014 IEEE Symposium on Security and Privacy, 2014, pp.639-654.

[15] S. K., Pasupuleti, *et al*., "An efficient and secure privacy-preserving approach for out sourced data of resource constrained mobile devices in cloud computing," *Journal of Network and Computer Applications*, Vol.64, pp.12-22, 2016.

[16] L Raja, *et al*., "An overview of MANET: applications, attacks and challenges," *International journal of computer science and mobile computing*, Vol.3, No.1, pp.408-417, January-2014.

[17] H. A. Shakir, *et al*., Enhancement of energy control routing protocol for mobile ad hoc network based on hybrid particle swarm optimization with ant colony-based energy control routing," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, Vol.8, No.2, pp.308-314, 2017.

[18] R. Gupta, "Mobile Ad-hoc network (MANETS) : Proposed solution to security related issues," *Indian J. Comput. Sci. Eng.*, Vol.2, No.5, pp.738746, 2011.

[19] G. Kirby, *et al*., "An approach to Ad-hoc cloud computing," *Eprint arXiv:*1002.4738, Feb.2010.

[20] A. Khan,*et al*.,"Towards secure mobile cloud computing: A survey," *The International Journal of eScience*, Vol.29, No.3, pp.1278-1299, 2013.

[21] Z. Xia, *et al*., "A Secure and Dynamic Multi-Keyword Ranked Search Scheme over Encrypted Cloud Data,". *IEEE Transactions on Parallel and Distributed Systems*, Vol.27, No.2, pp.340-352, 2016.

[22] B. H. Bloom, ”Space/time trade-offs in Hash Coding with Allowable Errors,” *in Communications of the ACM*, Vol.13, Issue 7, 1970.
[23] B. Chor, *et al*., ”Private information retrieval,” *J. ACM*, Vol.45, pp.965-981, 1998.
[24] J. Groth, *et al*., ”Multi-query computationally-private information retrieval with constant communication rate,” *In PKC*, 2010, pp.107-123.
[25] S. Kamara and C. Papamanthou, ”Parallel and dynamic searchable symmetric encryption,” *in Financial Cryptography and Data Security. Springer*, 2013, pp.258-274.
[26] C. Wang, *et al*., ”Enabling secure and efficient ranked keyword search over outsourced cloud data,” *IEEE Trans. Parallel Distrib. Syst*, Vol.23, No.8, pp.1467-1479, 2012.
[27] N. Cao, *et al*., ”*Privacy-preserving multi-keyword ranked search over encrypted cloud data*,” In Proceedings of IEEE INFOCOM. IEEE, Shanghai, China, 2011, pp.829-837.
[28] N. Cao, *et al*., ”Privacy preserving multi-keyword ranked search over encrypted cloud data,” *Parallel and Distributed Systems, IEEE Transactions on*, Vol.25, No.1, pp.222-233, 2014.
[29] C. Orencik, *et al*., ”*A practical and secure multi-keyword search method over encrypted cloud data*,” in Cloud Computing (CLOUD), IEEE Sixth International Conference on IEEE, 2013, pp.390-397.
[30] W. Zhang, *et al*., ”*Secure ranked multi-keyword search for multiple data owners in cloud computing*,” in Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on. IEEE, 2014, pp.276-286.
[31] B. Wang, *et al*., ”*Inverted Index Based Multi-Keyword Public-key Searchable Encryption with Strong Privacy Guarantee*,” in IEEE Conference on Computer Communications (INFOCOM), 2015, pp.2092-2100.
[32] C. Chen, *et al*., ”An Efficient Privacy-Preserving Ranked Keyword Search Method,” *IEEE Transactions on parallel and distributed systems*, Vol.27, No.4, pp.951-963, 2016.
[33] Z. Sherin and M. K. Soni, ”Biometric Stationed Authentication Protocol (BSAP) Inculcating Meta-Heuristic Genetic Algorithm,” *I.J. Modern Education and Computer Science*, 2014, pp.28-35.
[34] Z. Sherin and M. K. Soni, ”A Novel CryptBiometric Perception Algorithm to Protract Security in MANET. Genetic Algorithm”. *I.J. Computer Network and Information Security*, 2014, pp.64-71.
[35] Z. Sherin and M. K. Soni, ”*Secure Routing in MANET through Crypt-Biometric Technique*,” Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), 2014, pp.713-720.
[36] S. Lee, *et al*., ”OrtegaCompressed sensing and routing in multi-hop networks,” *CENG technical report University of Southern California*, 2009.
[37] D. A. Huffman, ”*A Method for the Construction of Minimum Redundancy Codes*,” Proceedings of the IRE, 1952.
[38] A. Setia and P. Ahlawat, ”*Enhanced LZW Algorithm with Less Compression Ratio*,” Proceedings of ICAdC, AISC 174, Springer India, 2013.
[39] HP Company, ”HP Case Study: WAN Link Compression on HP Routers,” 1995.
[40] V. Jacobson, ”Compression TCP/IP for Low-Speed Serial Link,” RFC 1144, 1990.
[41] M. Degermark, *et al*., ”IP Header Compression,” RFC 2507, 1999.
[42] A. Singhal, ”Modern information retrieval: A brief overview,” *IEEE Data Engineering Bulletin*, Vol.24, pp.35-43, 2001.
[43] D. Boneh and M. Franklin, ”Identity-Based Encryption from the Weil Pairing,” *in CRYPTO 2001, LNCS, 2139, Springer-Verlag*, 2001, pp.213-229.
[44] H. Dobbertin, *et al*., ”Advanced Encryption Standard-AES”, *Bonn, Germany, May 10-12, 2004 : Revised Selected and Invited Papers, LNCS, Springer*, 2005.
[45] C. Gu and Y. Zhu, ”New Efficient Searchable Encryption Schemes from Bilinear Pairings,” *International Journal of Network Security*, Vol.10, No.1, PP.25-31, 2010.
[46] R. Zhao, *et al*., ”*Privacy-preserving Personalized Search over Encrypted Cloud Data Supporting Multikeyword Ranking*,” Sixth International Conference on Wireless Communications and Signal Processing (WCSP),2014.