Peer-to-Peer Networking and Applications (2020) 13:2342–2355
https://doi.org/10.1007/s12083-020-00912-7

A dynamic and verifiable multi-keyword ranked search scheme in the P2P networking environment

Haoyang Wang[1] · Kai Fan[1] · Hui Li[1] · Yintang Yang[2]

Received: 11 October 2019 / Accepted: 26 March 2020 / Published online: 26 May 2020

Abstract

With the rapid development of the Internet, Peer to Peer(P2P) network has been applied in various fields. Users in P2P network also have a large amount of data, but users cannot provide enough storage space locally. More and more users choose to upload their own data to cloud server in order to save overhead and facilitate sharing their own data with other users. In order to ensure data security, researchers have proposed searchable encryption(SE) technology, and searchable encryption has been widely used. In this paper, a dynamic verifiable multi-keyword ranked search scheme is proposed under the background of P2P network and cloud storage service(CSS). On the basis of using secure kNN algorithm to encrypt index and traditional inner product algorithm to obtain ranked results, the scheme in this paper realizes forward and backward security by changing the structure of file vector and using modular residual computation. Meanwhile, the integrity and freshness of search results are verified by combining timestamp chain and Merkle tree. Finally, the security of this scheme under two threat models is analyzed, and the performance evaluation experiment is carried out on the document set.

Keywords P2P network · Secure kNN algorithm · Tree-based index · Time-stamp chain · Merkle tree · Forward and backward security

1 Introduction

In recent years, P2P network and application have developed rapidly, but the limited local storage space of users has become a bottleneck for their development, With the unique advantages of CSS, more and more users storing data in the cloud server. This reduces local storage and computational overhead while also enabling rapid development of SE technology. The efficiency of the SE scheme is affected by various factors such as index, trapdoors and encryption methods. Since Song et al. [21] first proposed a symmetric searchable encryption(SSE), the SE technology has made tremendous progress in improving search efficiency. Simultaneously, the cloud servers in most current SE schemes are set to be honest but curious. But in reality, malicious servers will perform illegal operations on the ciphertexts on them. Researchers have proposed some verifiable searchable encryption(VSE) scheme on this basis. However, most VSE schemes currently only supports static databases or based on specific SE structures. In the meantime, for most existing searchable encryption schemes that support dynamic updates, the update operations are performed on the cloud server, thus forward and backward security is very necessary for dynamic searchable encryption(DSE) schemes. They are defined as follows:

1. **Forward privacy**: If we search for a keyword and later add a new document containing keyword, the cloud server does not learn that the new document has a keyword we searched in the past.

This article is part of the Topical Collection: *Special Issue on Security and Privacy in Machine Learning Assisted P2P Networks* Guest Editors: Hongwei Li, Rongxing Lu and Mohamed Mahmoud

✉ Kai Fan
kfan@mail.xidian.edu.cn

Haoyang Wang
why19970701@163.com

[1] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China

[2] Key Lab. of the Minist. of Educ. for Wide Band-Gap Semicon. Materials and Devices, Xidian University, Xi'an, China

2. **Backward privacy**: New queries cannot be executed over deleted documents.

In this paper, we propose a dynamic and verifiable multi-keyword ranked search scheme in the P2P networking environment with forward and backward security(DVMRS). DVMRS can be summarized as follow:

1. DVMRS realizes forward and backward security in line with the traditional inner product algorithm and secure kNN algorithm [16, 17, 28], which are challenging issues on the past DSE schemes.
2. DVMRS combines Merkle tree and time-stamp chain to ensure verification of the search results integrity and freshness in the dynamic databases.

We organize this paper as follows: In Section 2, we give the related work on SE. And in Section 3, we introduce the design goals and preliminaries. Meanwhile, the system model and security model are described in Section 4. We present our scheme in Section 5 completely. Then we carry out the performance and security analysis in Section 6 and Section 7, respectively. Finally, we reveal the conclusion and future work of our scheme in Section 8.

2 Related work

Song et al. [21] first proposed a searchable symmetric encryption scheme using sequential for single keyword search on encrypted data, which proved to be safe. Then Boneh et al. [4] proposed a searchable encryption scheme with public key. Based on these two schemes, the research on searchable encryption has made great progress in recent years.

Multi-keyword search After a great quantity researches, [2, 5, 10] realized the linked multi-keyword search. Boneh et al. [5] proposed a public key based join and unjoin query scheme, which is similar to subset and range queries. Wang et al. [2] designed a public key scheme based on inverted index. This approach uses private set intersection to support joined multi-keyword queries. Wu et al. [27] constructed an efficient multi-keyword public key searchable encryption scheme based on the reverse encryption index structure and homomorphic encryption.

Ranked search Ranking search is proposed to sort the search results according to a certain method of relevance, so that search users can find more relevant search results faster. Wang et al. [24] uses inverted indexes and constructs a preserving order symmetric encryption scheme, but this scheme only supports single-keyword search. Cao et al. [19] first construct a searchable encryption scheme(MRSE) that supports multi-keyword ranked search using secure inner product computation with lower computing and communication overhead. However, this scheme ignored the importance of different keywords. Fu et al. [8] used processing stemming algorithm, LSH and bloom filter built a supports ranked and fuzzy search scheme at the same time. Sun et al. [22] extended the index structure to multi-dimensional binary(MDB) tree structure based on the Fu et al. [8]. Dai et al. [3] proposed a multi-keyword ranking search scheme for privacy protection of encrypted data in a hybrid cloud based on binary tree and keyword partition algorithm in line with an equally divided $k - means$ clustering. Zhang et al. [33] has modified the computation of term frequency(TF) value, which the words position and file length in the file are added into it.

Dynamic search In practice, the files stored by a cloud server are not immutable, so file updates should be considered. In Goh et al. [9], file update is implemented through bloom filter. Kamara et al. [14] constructed a new dynamic encryption index to implement dynamic updates.The scheme proposed by Wang et al. [25] implements efficient dynamic indexing under homomorphic encryption and pseudo-random padding. Wan and Deng et al. [23] utilized the bilinear-map accumulation tree to achieve updating scheme. Du et al. [6] put forward a dynamic multi-client searchable encryption scheme that supports boolean queries by incorporating the client authorization information into the query token and index.

Verifiable search Kamara et al. [13] firstly raised VSSE scheme in static databases. Wen and Deng et al. [23] proposed the verification application by homomorphic MAC. Wang et al. [26] proposed an efficient verifiable keyword searchable encryption utilizing aggregate keys to save computational resources. Zhang et al. [33] combined the binary tree and the Merkle tree based on inner product computation to implement the verification of data integrity.

3 Preliminaries

– **Incremental hash (IH)**. Incremental hash was first proposed by Bellare et al. [1] and was used by numerous existing SE schemes. An IH function is a collision-resistant function:

$$IH : \{0, 1\}^* \rightarrow \{0, 1\}^l \tag{1}$$

with which the addition or subtraction operation of two random strings in the IH function will not produce a collision.

– **Merkle tree**. Merkle tree was first proposed by kamara et al. [18] for the first time. The Merkle tree in DVMRS

is established on the basis of tree index. Firstly, the IH value corresponding to the file at each leaf node of the tree is calculated. Then the IH value of the non-leaf node is calculated by the concatenation operation by using the left and right child nodes of a non-leaf node, where || represents concatenation operation. And the IH value of the root node rt is obtained by repeating this process. In Fig. 1, we demonstrate how to build a Merkle tree.

- **Time-stamp chain**. The time-stamp chain is a chain consisting of fixed update time-stamps up_i, data update time-stamp $TP(i, j)$, and query time-stamp T_i, where the fixed update time-stamp is the point in time after a fixed period of time for updating, the data update time-stamp is the time point when data are updated and the query time-stamp is the point in time when the user queries. In Fig. 2, we depict the specific composition of a time-stamp chain in DVMRS.
- **Minimum hash sub-tree(MHS)**. The MHS is a part of Merkle tree, which we utilize the MHS to assist in calculating the root IH value of the search results. Take Fig. 1 for example, the request document is f_5. The MHS returned is consists of $hash22$ node and its two child nodes. When search user delivers a search query to the CSP, the CSP sends back the final search results along with MHS and the corresponding auxiliary information, which will be introduced in Section 5.1.

4 System design

4.1 Design goals

In order to realize the privacy protection, efficient search, multi-keyword ranked search and search results verification proposed in our scheme, we put forward the following design goals:

- **Search efficiently**: The index structure in SE scheme has a significant effect on search efficiency. In our scheme, tree-based index is a great to realize it.
- **Dynamic update**: Our scheme should support dynamic updating of data. When data owners increase, delete and update local data, we need to execute the same operation instructions for data on cloud server to ensure the consistency of local data between the CSP and data owner.
- **Privacy**: In this scheme, we need to realize data privacy, index and query privacy, forward and backward privacy, keyword privacy and trapdoor unlinkability. (1) Data Privacy. The CSP cannot conclude the corresponding plaintext by analyzing the stored ciphertext. (2) Index and Query Privacy. Both the query and index are represented by vectors. The index vector contains information such as TF value, while the query vector contains information such as inverted document frequency(IDF) value, so it is necessary to protect the privacy of them. (3) Keyword Privacy. The CSP could not distinguish the specific keywords.(4) Trapdoor Unlinkability. The trapdoor need to be distinguishable for the same query. (5) Forward and Backward Privacy. In dynamic schemes, forward and backward privacy are such an indispensable demand that should be implemented.
- **Results verification**: In DVMRS, we need to verify the completeness, freshness and correctness of the search results. (1) Completeness. The CSP returns the search results completely. (2) Freshness. The results returned to the user should be up to date. (3) Correctness. The CSP cannot return search results containing irrelevant data to the users.
- **Multi-keyword search**: Compared with single-keyword search, multi-keyword search is more able to meet the search requirements of data users and implement higher search efficiency.

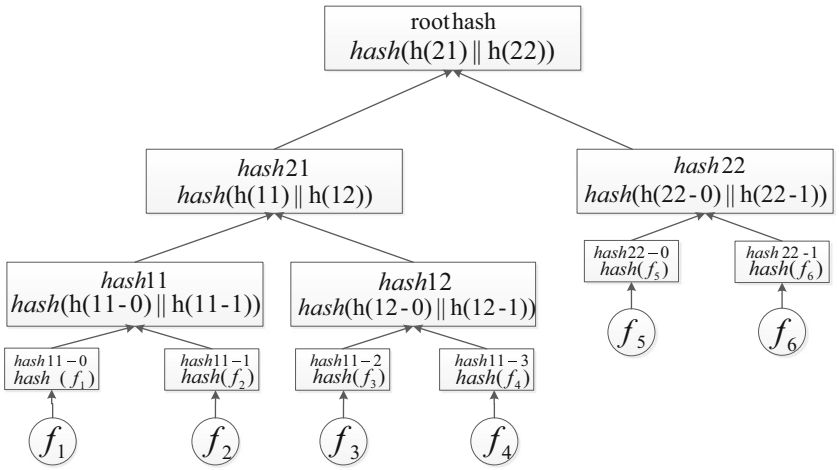

Fig. 1 Merkle tree

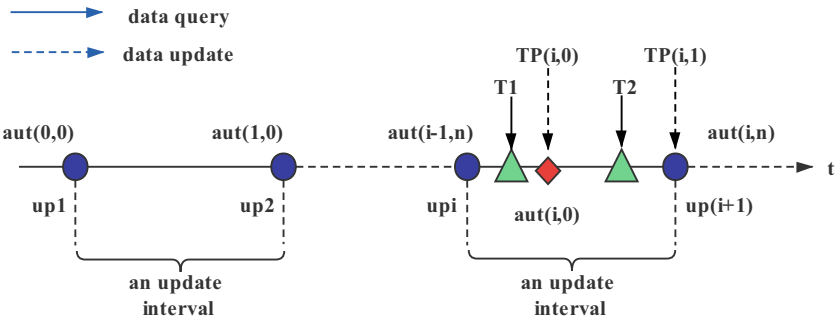

Fig. 2 Time-stamp chain

4.2 System model

As shown in Fig. 3, DVMRS consists of three entities.

- **Data owner**: The data owner will use symmetric encryption algorithm to encrypt the local data and outsourced them to the cloud service provider(CSP). Meanwhile, the data owner will use the data dictionary to build a tree-index structure, which will be transmitted to the CSP after encryption. The data owner sends the key that generates the search trapdoor and the symmetric key that encrypts the data to the authorized users through the secret channel.
- **Cloud service provider(CSP)**: The CSP is an intermediate entity that stores encrypted data and corresponding indexes obtained from the data owner and provides data access and search services to authorized data users. When an authorized user sends a trapdoor to the CSP, it returns a matching set of ciphertexts in line with trapdoor and the authenticator under the query time-stamp to the data user for results verification.
- **Data users**: An authorized user can obtain symmetric key and secret key from the data owner. When data users want to search on the CSP, firstly they need to generate a searching keyword set, then use the secret key to generate the corresponding trapdoor and send it to the CSP. The data user verifies the searching results by obtaining auxiliary information from the CSP.

4.3 Security model

In DVMRS model, the data users are authorized by the data owner to conduct the search, so we do not consider the leakage of the data users. Meanwhile, communication between data owner and data user is conducted through secret channel, so we do not need to consider in the communication between them.

However, the CSP is considered curious and dishonest in this scheme, the CSP may not follow the protocols and the CSP is curious about the data content, keywords and

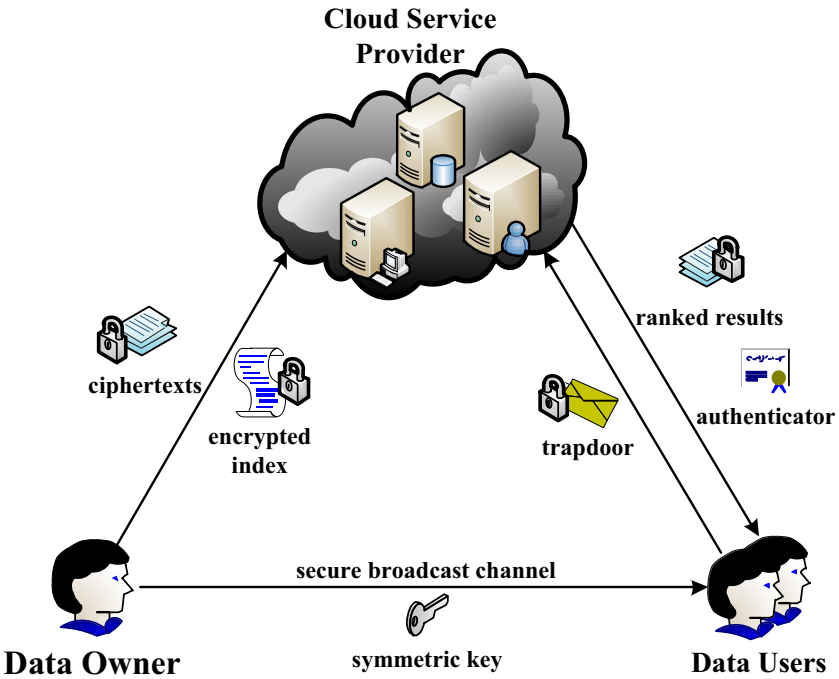

Fig. 3 System model

other additional information. DVMRS mainly considers two threat models:

- **Known Ciphertext Model**: In this model, the CSP is only aware of encrypted information, which includes encrypted data, encrypted indexes and searching trapdoors.
- **Known Background Model**: In this model, the CSP know information other than encrypted information, such as document frequency and keyword frequency. This information will be used for statistical attacks to infer keywords in query requests.

5 Scheme construction

5.1 Notations

Table 1 lists the symbols that need to be used in this scheme and the corresponding instructions.

5.2 Algorithm construction

5.2.1 Data owner

The following algorithms are executed by Data owner:

- $KGen(1^k) \rightarrow \{K_1, K_2, K_3, ssk, spk\}$: This algorithm is a probabilistic algorithm run by the data user, It takes a security parameter as input, and outputs the secret keys and a random signing key pair (ssk, spk). K_1 is used to encrypt the index based on tree construction, K_2 is used to encrypt documents and K_3 is used to generate authenticator with the of ssk. The secret key $K_1 = (S, M_1, M_2)$. S is an $(m + d + 1)$-bit vector, M_1 and M_2 are two $(m + d + 1) \times (m + d + 1)$ invertible matrices respectively. The vector S is a splitting indicator to split document vector into two random vectors, which confuses the value of document vector, M_1 and M_2 are used to encrypt the splitting vectors simultaneously.
- $Enc(F, K_2) \rightarrow \{C\}$: The data owner uses secret key K_2 to generate the ciphertexts of document set.
- $GenIndex(F, K_1, K_3, ssk) \rightarrow \{I, \pi\}$: This algorithm has divided into two sub-algorithms $BuildTree(F, K_1) \rightarrow \{I\}$ and $GenMerkleTree(I, K_3, ssk) \rightarrow \{\pi\}$. The data owner generates an m-bit vector P_j according to each document $F_j (j = 1, 2, \cdots, N)$, where each bit $P_j[i]$ indicates the weight value of keyword t_i in F_j i.e. $P_j[i] = Value(t_i, F_j)$. Then the data owner extends the vector P_j to an $(m+d+1)$-bit vector $P'_j = P_j \,||\, V_j$, where V_j is a $(d + 1)$-bit vector. To achieve certain scalability of document collection, d should be set as possible maximum size of the outsourced document collection. Assume the possible maximum similarity score between a search keyword set and a document is $\max_s$, then the data owner chooses a random parameter g, where $g > \max_s$. Therefore , the vector $V_j (j = 1, 2, \cdots, N)$ can be generated as shown in Algorithm 1, where g^* represents arbitrary integer multiples of g. Thus, the type of V_j is as follows:

$$\begin{aligned} V_1 &= (g^*, g^*, \cdots, a_1, g^*, \cdots, g^*, -a_1) \\ V_2 &= (g^*, g^*, a_2, \cdots, g^*, \cdots, g^*, -a_2) \\ V_3 &= (g^*, g^*, \cdots, g^*, a_3, \cdots, g^*, -a_3). \end{aligned} \tag{2}$$

Algorithm 1 Generation algorithm of V_j.

Input: number sets σ and $\bar{\sigma}$
Output: vector set $(V_1, V_2, \cdots, V_N)$, updated number sets σ' and $\bar{\sigma}'$

1: **for** $j = 1$ to N **do**
2: random choose an element v_j from $\sigma : v_j \leftarrow_R \sigma$
3: random choose $a_j \leftarrow_R Z_q^*$
4: $V_j = \{V_j[i]\} = \begin{cases} a_j (i = v_j) \\ -a_j (i = d + 1) \\ g^*(other) \end{cases}$
5: $\sigma = \sigma - \{v_j\}$
6: $\bar{\sigma} = \bar{\sigma} + \{v_j\}$
7: **return** $(V_1, V_2, \cdots, V_N), \sigma', \bar{\sigma}'$
8: **end for**

The vector P'_j will be encrypted by the secure kNN algorithm: the data owner uses vector S to split P'_j into two $(m + d + 1)$ dimensional vectors (p_a, p_b), where the vector S functions as a splitting indicator. Namely, if $S[i] = 0 (i = 1, 2, \cdots m + d + 1)$, $p_a[i]$ and $p_b[i]$ are both set as $P'_j[i]$; if $S[i] = 1 (i = 1, 2, \cdots m + d + 1)$, the value of $P'_j[i]$ will be randomly split into $p_a[i]$ and $p_b[i]$ i.e. $P'_j[i] = p_a[i] + p_b[i]$. Then the index of encrypted document C_j can be calculated as $I = (M_1^T p_a, M_2^T p_b)$. Moreover, according to the Merkle tree construction method in Section3, the data owner builds the relevant Merkle tree M_I in line with the index tree. Finally, a key tuple $(K_1, K_2, \bar{\sigma})$ will be sent to the authenticated search users through secure broadcast channel, where K_2 is the symmetric key used to encrypt documents outsourced to the CSP. And the data owner publishes g and stores σ in her own storage space. Meanwhile, the data owner outputs the authenticator π, stores the tree index and the original authenticator locally, and then sends them to the CSP.

- $PreUpdate(K_1, K_2, K_3, ssk, f) \rightarrow \{\tau_u, \pi_{i,j}\}$: This algorithm run by the data owner. It takes as input the symmetric keys K_1, K_2, K_3, the secret key of signing

key pair ssk, a file f to be updated and outputs the update token τ_u and the new authenticator $\pi_{i,j}$. The data owner sends τ_u and $\pi_{i,j}$ to the CSP.

– $Update(I, \tau_u, \sigma, \bar{\sigma}) \rightarrow \{I', \sigma', \bar{\sigma}'\}$ This algorithm is used to perform the update token. It takes the tree index, the update token τ_u, the previous number sets σ and $\bar{\sigma}$ as input, then outputs the new tree index and updated number sets σ', $\bar{\sigma}'$. The Merkle tree will be updated simultaneously. After the execution of this algorithm, the data owner delivers the new index and number sets to CSP.

5.2.2 CSP

The following algorithms are executed by CSP: $Search(I, TD) \rightarrow \{R, \pi_q^t, \pi_{i,j}\}$: This algorithm uses index and trapdoor to calculate the similarity score to get the $top-k$ searching results. After receiving the trapdoor TD from the data users, the CSP calculates the similarity score between TD and the index vector stored in each node to get the top-k relevant results. The similarity score is calculated as:

$$\begin{aligned} Score(\bar{W}, F_j) &= I \cdot TD \\ &= (P_j' \cdot Q_{\bar{W}}')modg \\ &= (p_a \cdot q_a + p_b \cdot q_b)modg \\ &= (M_1^T p_a, M_2^T p_b)\cdot(M_1^{-1}q_a, M_2^{-1}q_b)modg \\ &= (P_j \cdot Q_{\bar{W}})modg \end{aligned} \tag{3}$$

The larger score indicates the corresponding document F_j is more relevant to the search keyword set $\bar{W}$, the documents with top scores will be returned to the data user. Meanwhile, the CSP returns the authenticator π_q^t to the requested user when the data user sends a request to server and the authenticator $\pi_i, _j$ of the check point.

Moreover, It should be noted that when calculating the similarity scores in the first step, it is not necessary to calculate the similarity scores on each node one by one. When the weight of the corresponding keyword position on a non-leaf node is zero, we directly give up searching on the child nodes of this node,which can decrease amount of communication and computational overhead.

5.2.3 Data user

The following algorithms are executed by data user:

– $GenTrapdoor(\bar{W}, K_1, \bar{\sigma}) \rightarrow \{TD\}$: The data user generates the keyword set $\bar{W}$ for searching. Then data user creates an m-bit vector $Q_{\bar{W}}$ according to $\bar{W}$, where $Q_{\bar{W}}[i]$ indicates whether the i-th keyword of dictionary w_i is in $\bar{W}$ i.e. $Q_{\bar{W}}[i] = 1$ indicates yes and $Q_{\bar{W}}[i] = 0$ indicates no. Then the data user also extends the vector $Q_{\bar{W}}$ to an $(m + d + 1)$-bit vector $Q'_{\bar{W}} = Q_{\bar{W}}||V'$, where V' is a $(d + 1)$-bit vector and can be generated as follows: $V' = \{V'[i]\} = 1(i \in \bar{\sigma} \cup \{d + 1\})$ or $V' = \{V'[i]\} = 0(other)$. The data user can split $Q'_{\bar{W}}$ into two $(m + d + 1)$-bit vectors (q_a, q_b): if $S[i] = 0(i = 1, 2, \cdots, m + d + 1)$, the value of $Q'_{\bar{W}}$ will be randomly split into $q_a[i]$ and $q_b[i]$; if $S[i] = 1(i = 1, 2, \cdots, m + d + 1)$, $q_a[i]$ and $q_b[i]$ are both set as $Q'_{\bar{W}}[i]$. Thus the trapdoor TD can be generated as $TD = (M_1^{-1}q_a, M_2^{-1}q_b)$.

– $Dec(R, K_2) \rightarrow \{PR\}$: The data user uses the secret key K_2 transmitted from the data owner by a secure broadcast channel to decrypt the encrypted results R and get the plaintext results PR.

– $Check(K_3, spk, \pi_q^t, \pi_{i,j}) \rightarrow \{b_1\}$: The data user runs this algorithm to check the correctness of authenticator, which they get at query time t, this algorithm takes symmetric key K_3, the public key of signing keypair spk, authenticator π_q^t and $\pi_{i,j}$ as input. It outputs a bit b_1 represents an accept or reject result. The detailed process of this algorithm is shown in Algorithm 2:

Algorithm 2 Check the authenticator π_q^t.

Input: $K_3, spk, \pi_q^t, \pi_{i,j}$
Output: $b_1 \in \{0, 1\}$
let $\pi_q^t = \{\alpha_q^t, Sig_q^t\}$ and $\pi_{i,j} = \{\alpha_{i,j}, Sig_{i,j}\}$
if then $\alpha_q^t \neq (Sig_q^t)_{spk} || \alpha_{i,j} \neq (Sig_{i,j})_{spk}$
  return $b_1 = 0$
end if
$(rt_q^t, tp_q^t, \alpha) \leftarrow Dec_{K_3}(\alpha_q^t)$
if then tp_q^t is not before the previous update time point
  $\alpha_k = \alpha_{i,j}$
  for do $\alpha_k \neq \emptyset$
    $(rt_k, tp_k, \alpha_{k-1}) \leftarrow Dec_{K_3}(\alpha_k)$
    if then $tp_k < t$
      break
    end if
    let $\alpha_k = \alpha_{k-1}$
  end for
  if then $\alpha_k = \alpha_q^t || \alpha_k = \emptyset$
    return $b_1 = 1$
  else
    return $b_1 = 0$
  end if
else
  return $b_1 = 0$
end if

– $Verify(K_1, K_3, spk, R, \pi_q^t, aux) \rightarrow \{b_2\}$: The data users run this algorithm, which takes symmetric keys

K_1, K_3, the public key of signing key pair spk, the searching results R, the authenticator π_q^t and the corresponding auxiliary information aux returned by the CSP. It outputs a bit b_2 represents verification passed or not. The verification process of searching results is shown in algorithm 3:

Algorithm 3 Verify the authenticator π_q^t.

Input: K_1, K_3, spk, R, π_q^t, the corresponding auxiliary information aux, returned results R
Output: $b_2 \in \{0, 1\}$

```
if t thenhe authenticator of root node is true
    if v thenerification of each node in mintree is true
        recomputing the hash value of root with mintree and aux
        if t thenhe value after recomputing = the value in authenticator
            re-search the mintree using the same trapdoor
            if r thene-search results = R
                return b2 = 1
            else
                return b2 = 0
            end if
        end if
    end if
end if
```

5.3 Detailed process

5.3.1 Anthenticator construction

We utilize the Merkle tree and time-stamp chain to design authenticator. There are two challenges in verifying searching results.

The first one is how to design an efficient generic proof index not only supports data integrity verification but also supports data update. In DVMRS we build and maintain such a proof index by leveraging the fully Merkle tree and IH values.

The second one is how to ensure data freshness by preventing the root from being replayed(malicious server) in the context of data updates. In order to solve this problem, we add a time-stamp to the authenticator, which is located on the three-party known time-stamp chain, it ensures that each authenticator will have a time identity that can not be tampered with.

Combined with the solutions to the above two problems, we design the authenticator used in the scheme as follows:

$$\begin{cases} \pi_{i,0} = (\alpha_{i,0}, Sig_{ssk}(\alpha_{i,0})), (up_i < tp_{i,0} \le up_{i+1}) \\ \alpha_{i,0} = Enc_{K_3}(rt_{i,0}||tp_{i,0}) \\ \cdots \\ \pi_{i,j} = (\alpha_{i,j}, Sig_{ssk}(\alpha_{i,j})), (tp_{i,j-1} < tp_{i,j} \le up_{i+1}) \\ \alpha_{i,j} = Enc_{K_3}(rt_{i,j}||tp_{i,j}||\alpha_{i,j-1}) \\ \cdots \\ \pi_{i,n} = (\alpha_{i,n}, Sig_{ssk}(\alpha_{i,n})), (tp_{i,n} = up_{i+1}) \\ \alpha_{i,n} = Enc_{K_3}(rt_{i,n}||tp_{i,n}||\alpha_{i,n-1}) \end{cases} \tag{4}$$

The detailed process of constructing authenticator:

1) Generating the IH hash values of files.
2) Building the Merkle tree based on the IH values of files.
3) Data owner determines a time-stamp chain, which the time-stamp chain contains the fixed update time point of authenticator and interval length of the update time.
4) Data owner generates a original authenticator and time-stamp chain, meanwhile broadcasts them among data user with broadcasting channel and upload them to CSP. It is noted that we use Network Time Protocol running on the CSP to synchronize the clocks among the data owners and the data users during their interactions with the servers.

In DVMRS, the update of authenticator is divided into two cases:

The first case is that the Merkle tree is not updated during an update interval, the data owner only needs to update the time-stamp at next time point.

The second case is that the document set of the data owner is modified within an update interval, which means the root of the Merkle tree has been updated, then the data owner will calculate a new authenticator by using the latest root IH value and present time-stamp, meanwhile the CSP need to update authenticator simultaneously.

5.3.2 Results verification

The process of data users verifying searching results divided into two stages:

1) In order to prevent malicious CSP from returning incorrect anthenticator to the user, we design *Check algorithm* to check the freshness and correctness of authenticator, it is executed by a data user and verifies whether the authenticator has been replayed.

Table 1 Notations summary

Notation	Instruction
F	The plaintext documents which data owners store locally, it includes N files $F = f_1, f_2, ..., f_N$.
W	The keywords set, data owners extracts n keywords from documents set, it includes W keywords $W = w_1, w_2, ..., w_n$.
$\bar{W}$	The searching keywords set is generated by the search users and it is subset of keywords set, it includes m keywords $\bar{W} = \bar{w_1}, \bar{w_2}, ..., \bar{w_m}$.
$Q_{\bar{W}}$	The query vector submitted by data users contains keywords existing in searching keywords set $Q_{\bar{W}} = q_1, q_2, \cdots, q_m$.
T	The keywords extracted by the data owners from the files build the index tree, which are stored locally by the data owners.
I	The encrypted index tree generated from the tree structure T.
TD	The trapdoors generated from the query vector Q_W and will upload to the CSP.
σ	A number collection, initialized as 1, 2, 3, ..., d, d is set as possible maximum size of the outsourced document collection.
$\bar{\sigma}$	a number collection, initialized as ϕ.
R	The top-k encrypted document searching results returned from the CSP for decryption and verification.
PR	The plaintext document results decrypted by R.
M_I	The Merkle tree produced by data owner in the light of index tree.
tp	This notation represents the time-stamp in our system.
rt	This represents the hash value of the Merkle tree root.
π_q^t	This notation indicates the authenticator obtained after the user executes the query operation.
$\pi_{i,j}$	This notation represents the authenticator in i-th update interval which i means i-th update interval and j represents the number of changes in data set And when arrives at next update point, the authenticator will be $\pi_{i+1,0}$.

2) We use *Generate algorithm* to verify searching results by leveraging the root hash value extracted from the authenticator and the minimum hash sub-tree extracted from Merkle tree with searching results.

In order to prevent the CSP from previous authenticators and ensure the freshness of the root hash value, the application of time-stamp chain can solve this problem, such that data users can trace authenticators in the chain and identify if root hash value is fresh.

In this setting, the CSP needs to provide an authenticator at the query time and meanwhile an authenticator at the checkpoint, where the checkpoint is referred as the next update time point close to the query time t.

When data users searching files with keywords in keywords dictionary, there have three cases at different points with same keywords, we have demonstrated these cases in Fig. 4 in sub-section 2.5:

1) The first case is that the query occurs at t_1, the malicious CSP could only send $\pi_{i-1,n}$ to the data user.
2) The second case is that the query occurs at t_2 after the data update at $tp_i,_0$, and the authenticator that the malicious CSP sends to the user is $\pi_i,_0$.
3) The last case is that query is generated at t_2, and the authenticator sent by the malicious CSP is $\pi_{i-1,n}$. In the last case, a data freshness attack occurs, but it will be detected at the checkpoint up_{i+1}. The data user will obtain the last authenticator $\pi_i,_1$ from the malicious CSP at the checkpoint to verify whether the data obtained at the query time has been replayed or not.

6 Performance analysis

The DVMRS is to improve the security on the secure kNN neighbor-based searchable encryption scheme. By changing the structure of constructing the index tree and generating the trapdoor vector, the scheme adds a number set to mark the data to achieve forward and backward security.

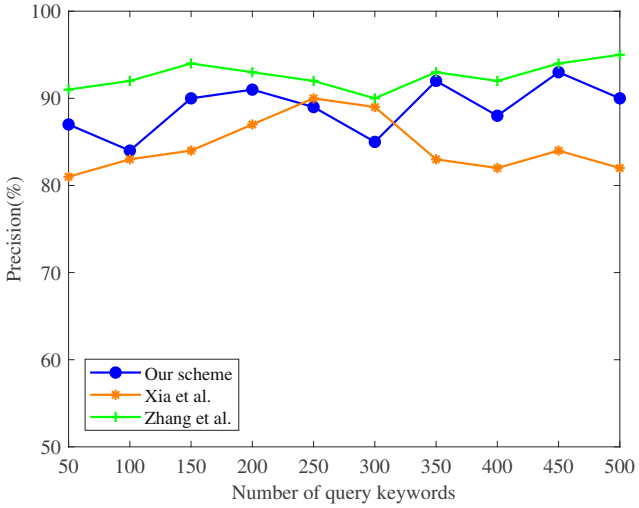

Fig. 4 Precision test

Verification of data integrity, freshness and correctness is achieve by combining time-stamp chain and Merkle tree.

6.1 Precision

In the searching accuracy analysis, we use the following formula to define the precision:

$$Precision = \frac{k'}{k} \tag{5}$$

where k' is the number of real documents and k is the number of returned top-k documents.

The schemes for comparison with this scheme are Xia et al. [16] and Zhang et al. [17], and the accuracy of this scheme is slightly higher than that of Xia et al. [16]. The scheme, Zhang et al. [17], is slightly higher than DVMRS and the Xia et al. [16], which is due to the modification of the inner product $TF \times IDF$ calculation. The method assigns different weights to search for keywords in different positions in the data, thereby making the search accuracy higher. However, DVMRS realizes the forward and backward security that is not realized by other schemes. This scheme also introduces the time-stamp chain and combines the Merkle tree to resolve it.The final three precision comparison simulation diagram is shown in Fig. 4.

In order to ensure the fairness of the simulation comparison, the same parameters and the same data set were used in the simulation comparison.

The data set used in the simulation contains 3026 documents and 1789 keywords extracted from the document. It can be seen from the simulation image that there is no obvious fluctuation in the search accuracy during the change of the number of keywords in the query from 50 to 500 between three schemes. As can be seen from Fig. 5, the DVMRS maintains high search accuracy while achieving forward and backward security.

6.2 Index construction

In both the DVMRS and two schemes involved in the comparison, the index construction is divided into two stages, building the tree structure and encryption, and $TF \times IDF$ algorithm and kNN nearest neighbor algorithm are used in both Zhang et al. [17] and Xia et al. [16] schemes. In the encryption stage, the two comparison schemes will extend the n-bit original vector to $(n + U + 1)$-bit, where U is a random number. However, we should note that U is a random number in a certain range. When there are more keywords in a file, the range of U random should be increased accordingly, otherwise its random security can not be brought into play.

In DVMRS, in order to achieve forward and backward security, we modify the vector filling method of vector to extend the original vector of n-bit to $(n + d + 1)$-bit, where d is the maximum value of the outsourced document set. In this way, the larger number of document sets, the more bits the vector accounts for, although it will increase a small amount of computing overhead, but the security will be stronger. Simultaneously, it is noted that in order to ensure the security of keywords, two schemes for comparison should be also adjust the range of U according to the quantity of stored data.

In Fig. 5, the number of keywords in the dictionary is fixed to n=750 for the number of different documents from 500 to 3000. In Fig. 6, the number of documents in the dictionary from 250 to 1500 is fixed to N=1500.

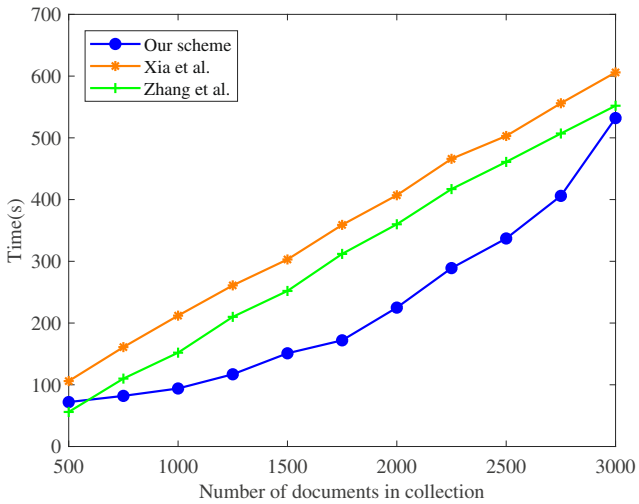

Fig. 5 Build index(a)

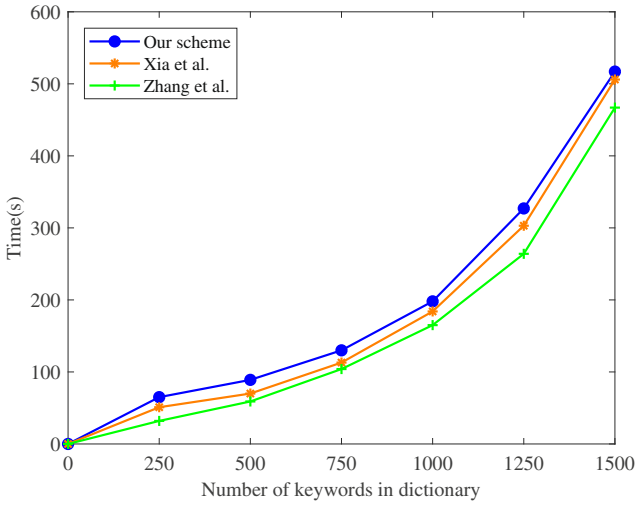

Fig. 6 Build index(b)

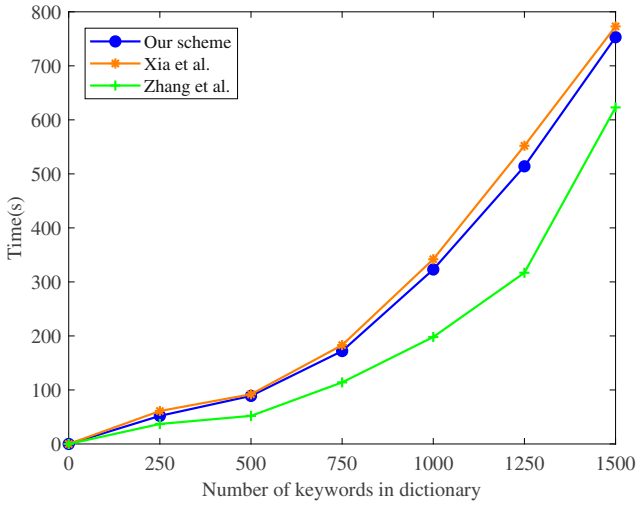

Fig. 7 Trapdoor Generation(a)

6.3 Trapdoor generation

In DVMRS and the contrast schemes, the complexity of trapdoor generation is determined by the segmentation vector and secret matrix used in encryption, and it is also related to the total number of keywords extracted from the document set. When the total number of keywords in the dictionary is n, the complexity of generating trap gate is $O(n^2)$. Meanwhile, the filling vector in encryption is improved in the DVMRS. The time complexity of trapdoor generation is also affected by the number of document sets, but in the acceptable range.

We have simulated the trapdoor generation many times, as shown in Figs. 8 and 9 respectively. So in practice, there will be a large amount of data in CSP computing environment. There will only be a small gap in the time complexity of trapdoor generation.

In Fig. 7, the number of keywords in each query is fixed at 10, and the number of keywords in the dictionary changes from 500 to 2000. The fixed keyword in the dictionary in Fig. 8 is n=500, and the number of words in each query is from 5 to 30.

6.4 Search efficiency

The search process of this scheme is as follows:(1) Calculate the inner product of the relevant node vector and the query vector in the tree;(2) Return the document with higher similarity score according to the number of returned documents and the obtained similarity score. Therefore, the searching time complexity is mainly affected by the number of nodes in the tree, i.e. mainly depends on the number of documents. When the number of stored documents is N, the time complexity is $O(log_2 N)$. It is also affected by the number of keywords in the dictionary. We compare the DVMRS with the schemes in Zhang et al. [17] and Xia et al. [16], where we simulate different cases of different documents and different numbers of keywords. The simulation results are shown in Figs. 9 and 10.

In Fig. 9, the number of keywords is fixed to $n = 500$, the number of search keyword set is $m = 10$, the number of returned documents is $k = 30$, and the number of documents changes from 500 to 3000. In Fig. 10, the number of documents is fixed to $N = 1000$, the number of search keyword set is $m = 10$, and the number of returned documents is $k = 30$. The number of keywords in dictionary varies from 250 to 1500.

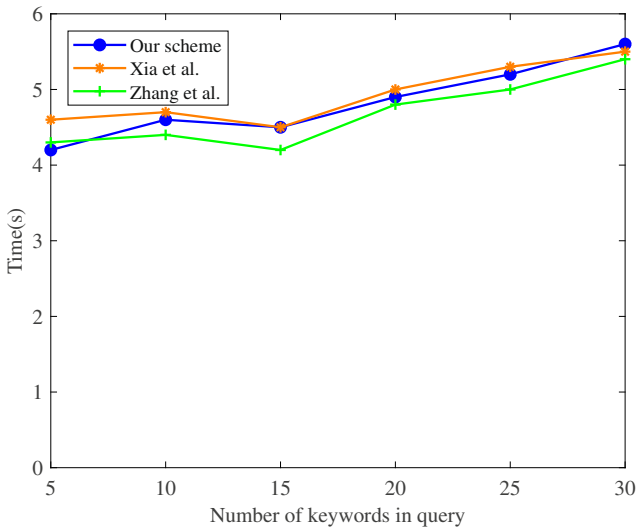

Fig. 8 Trapdoor Generation(b)

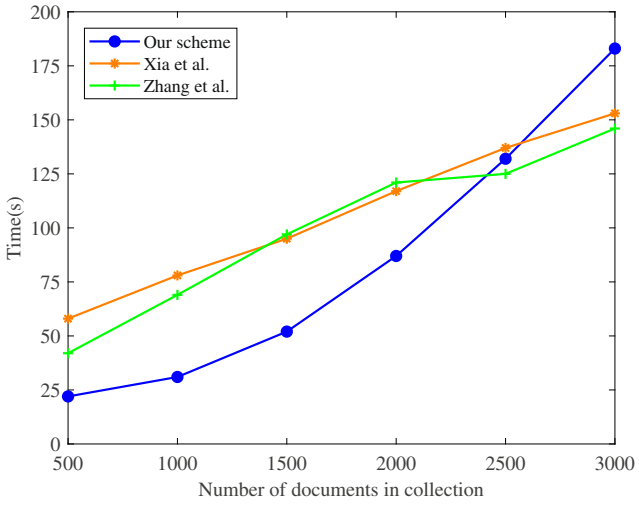

Fig. 9 Search Efficiency(a)

7 Security analysis

7.1 Privacy protection

1. **Data privacy**. The document set is encrypted locally by the data owner using symmetric encryption. The data owner directly uploads the ciphertexts to the CSP after encryption. The data owner transfers symmetric key to the authorized users through the secret channel. The data user is proven to be semantically secure through a symmetrically encrypted set of documents.
2. **Index and Trapdoor privacy**. In DVMRS, the security of the index and the trapdoor are based on the secure kNN algorithm. Although the keyword set of the two filters or the two search keyword sets are the same, the indexes or trapdoors are not the same. This is because the secure kNN algorithm is a non-deterministic algorithm. In this algorithm, the segmentation vector S used to split the trapdoors and indexes, two matrices M_1, M_2 are randomly generated. As long as the security of the encryption key K_1 is ensured, the CSP cannot identify the trapdoors and indexes by analyzing the plaintext. This has also proven to be safe in the context of Known Ciphertext Model.
3. **Trapdoor Privacy**. In constructing the vector $V_j (j = 1, 2, \cdots, N)$, the DVMRS first selects a random parameter $g (g >$ similarity score maximum value). Then selects a parameter g^* of any integer multiple of g to construct the vector V_j. The trapdoor vector is randomly divided into two vectors and then encrypted, which protects the search pattern and makes it indistinguishable from different trapdoors and even the same trapdoors. Therefore, the similarity score will be different for each query, and the CSP cannot distinguish them.
4. **Keyword Privacy**. A random number is used to randomize similarity scores when data owner encrypts indexes, thus the keyword privacy can be secured under Known Background Model.
5. **Forward and Backward Security**. Since the CSP is malicious in the context of DVMRS, some historical information such as previous queries and deleted documents are saved in the CSP local storage space.

   In this scenario our goal is that newly inserted documents cannot be searched by previous, or new queries cannot be executed on deleted documents. In DVMRS, a legitimate query calculates similarity scores as follows:

$$\begin{aligned} Score \quad & (\bar{W}, F_j) \\ &= (P_j' \cdot Q')modg \\ &= (P_j \cdot Q + \sum_{i=1}^{d+1} V_j[i] \cdot V'[i])modg \\ &= (P_j \cdot Q + a_j + g^* - a_j)modg \\ &= P_j \cdot Q \end{aligned} \tag{6}$$

   However, when previous queries (or new queries) "touch" new files (or deleted files), there will be:

$$\begin{aligned} Score \quad & (\bar{W}, F_j) \\ &= (P_j' \cdot Q')modg \\ &= (P_j \cdot Q + \sum_{i=1}^{d+1} V_j[i] \cdot V'[i])modg \\ &= (P_j \cdot Q + g^* - a_j)modg \\ &= (P_j \cdot Q - a_j)modg \end{aligned} \tag{7}$$

   Because the parameter a is randomly generated, the correct similarity score cannot be calculated. Therefore, this scheme has forward and backward security.

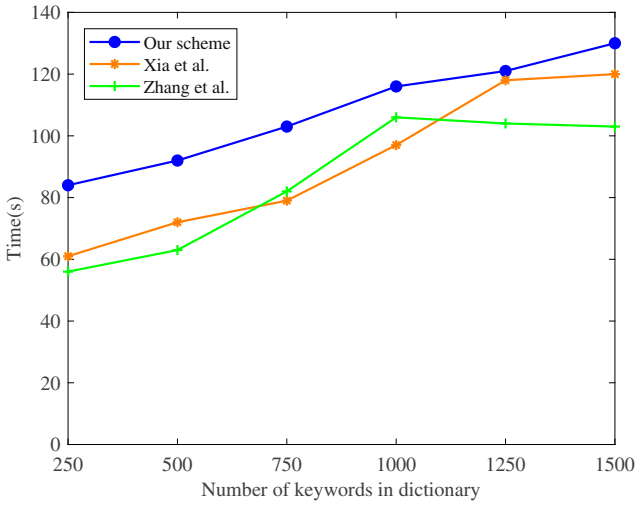

Fig. 10 Search Efficiency(b)

7.2 Security model

Known Ciphertext Model. In DVMRS, the CSP only knows the encrypted information, especially the encrypted document set C, the encrypted index tree I, and the trapdoor TD. The adversary could only distinguish between two files by "generated index" and "encrypted file". The vector representing the file is (m+d+1)-bit, the first m-bit represents the weight of the keywords, and the latter (d+1)-bit represents the dimension of the extended vector V.

In the index generation stage, we first expand the vector P_j corresponding to each file to P_j^*, and then use the segmentation vector to segment P_j^*. When $S[i] = 1$, $P_j^*[i]$ will be randomly divided into two vectors $P_a[i]$ and $P_b[i]$.Assuming that the number of "1" in the first m-bit is μ_1 and the dimension of each file is η_f, there are $(2^{\eta_f})^{\mu} \cdot (2^{\eta_f})^d$ possible combinations, while the two vectors are encrypted by the matrix of $(m + d + 1) \times (m + d + 1)$ dimensions. Assuming that each element is η_M-bit in the matrix, then the two matrices have $(2^{\eta_M})^{(m+d+1)^2 \times 2}$ possible values. Therefore the probability of the same index of two files is calculated as follows:

$$P_d = \frac{1}{(2^{\eta_f})^{\mu_1} \cdot (2^{\eta_f})^d \cdot (2^{\eta_M})^{(m+d+1)^2 \times 2}} = \frac{1}{2^{\mu_1 \eta_f + \eta_f d + 2\eta_M (m+d+1)^2}} \tag{8}$$

It can be seen from the above equation that the parameter μ_1, d, η_f, η_M are larger, the more difficult it is to distinguish, so the encrypted index is indistinguishable.

Known background model In DVMRS, the CSP may obtain other information, such as a file search frequency and a keyword search frequency, in addition to the encryption information. These information will be used for statistical attacks to infer the keywords in the query.

In DVMRS, the trapdoor is the vector of (m+d+1)-bit, and the former m-bit represents whether the keywords of the corresponding position exist in the query, the remaining (d+1)-bit vector represents $V' = \{V'[i]\} = 1(i \in \bar{\sigma} \cup \{d + 1\})$ or $V' = \{V'[i]\} = 0(other)$ in the extension vector. Firstly, the random number r ofη_r is used to extend the vector, which has 2^{η_r} possible values. Then the vector is divided into two vectors using the segmentation vector of (m+d+1)-bit, in which there are μ_0 "0". Assuming that each dimension of the former (m+d)-bit of the vector is η_q-bit, then there are a total of $(2^{\eta_q})^{\mu_0}$ possibilities. Then the random matrix is used to encrypt the two query vectors. The same probability of the two trapdoors is calculated as follows:

$$P_d = \frac{1}{2^{\eta_r} \cdot (2^{\eta_q})^{\mu_0}} \tag{9}$$

As can be seen from the above formula, when η_r, η_q, μ_0 are set to larger numbers, the indistinguishable query vector can be achieved.

8 Conclusion and future work

8.1 Conclusion

On the basis of a multi-keyword ranked scheme based on traditional inner product, we propose a searchable encryption scheme with forward and backward security and search results verification.

Firstly, we realize forward and backward security by changing the construction method represents the document vector and introducing two number sets.

Moreover, in order to make the verification of the returned results more secure and effective, we combine the Merkle tree and the time-stamp chain to ensure the data freshness, preventing the malicious server from returning the wrong authenticator. The DVMRS in this paper has also been extended to implement dynamic updates.

Finally, we analyze the performance and security of the proposed scheme. In the security analysis, we prove the two threat models proposed in the paper (Table 2).

The work done in this article still has space for improvement. In the model of DVMRS, the data owner sends the authenticator to the users through a secure broadcast channel, and also informs all users through it when the data is updated.

8.2 Future work

Above all, with the rapid development of machine learning(ML) and federated learning(FL), the security and privacy of data in them also need to be guaranteed [11, 30–32]. The combination of SE technology and ML,FL will be a research focus in the future, and it is the problem that we will continue to study as well.

Moreover, a variety of novel computing modes such as edge computing and fog computing have been proposed in recent years. These computing modes have been used in the design of SE schemes [7, 15, 20, 29] due to their high efficiency and low latency. DVMRS could try to combine these computing modes to improve query efficiency and reduce communication latency as well.

Eventually, the blockchain technology has developed mature nowadays, and its applications in the context of the Internet of Things are also increasing. Many desirable properties of the blockchain, such as decentralization, encryption technology and unchangeable transaction records, which make it have great applied value in privacy protection and authentication and key authorization of SE [12]. In the

Table 2 Scheme comparison

Scheme	Verification	Dynamism	Forward and Backward security	Privacy protection	$TF \times IDF$
Xia et al.	×	√	×	√	Tradition
Zhang et al.	√	√	×	√	Location and length
Ours	√	√	√	√	Tradition

subsequent work, we will also try to combine DVMRS with the blockchain to achieve fine-grained management of multi-user authority.

Acknowledgments This work is supported by the National Key R&D Program of China (No. 2017YFB0802300), the National Natural Science Foundation of China (No. 61772403 and No. U1836203), the Natural Science Foundation of Shaanxi Province(No.2019ZDLGY12-02), the Shaanxi Innovation Team Project(No.2018TD-007), the Xi'an Science and technology innovation plan(No.201809168CX9JC10) and National 111 Program of China B16037.

References

1. Bellare M, Goldreich O, Goldwasser S (1994) Incremental cryptography: the case of hashing and signing. Lecture Notes in Computer Science 839:216–233
2. Bing W, Wei S, Lou W, Hou YT (2015) Inverted index based multi-keyword public-key searchable encryption with strong privacy guarantee. In: Computer communications
3. Dai H, Ji Y, Yang G, Huang H, Yi X (2019) A privacy-preserving multi-keyword ranked search over encrypted data in hybrid clouds. IEEE Access
4. Dan B, Crescenzo GD, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. Eurocrypt 3027(16):506–522
5. Dan B, Waters B (2007) Conjunctive, subset, and range queries on encrypted data. Tcc 4392:535–554
6. Du L, Li K, Liu Q, Wu Z, Zhang S (2020) Dynamic multi-client searchable symmetric encryption with support for boolean queries. Inform Sci 506:234–257
7. Fan K, Xu H, Gao L, Li H, Yang Y (2019) Efficient and privacy preserving access control scheme for fog-enabled iot. Future Gen Comput Sys 99:134–142
8. Fu Z, Wu X, Guan C, Sun X, Ren K (2017) Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans Inform Forensics Secur 11(12):2706–2716
9. Goh E-J (2003) Secure indexes. Cryptology ePrint Archive, Report 2003/216. https://eprint.iacr.org/2003/216
10. Golle P, Staddon J, Waters B (2004) Secure conjunctive keyword search over encrypted data. In: Proceedings applied cryptography & network security conference
11. Hao M, Li H, Luo X, Xu G, Yang H, Liu S (2019) Efficient and privacy-enhanced federated learning for industrial artificial intelligence. IEEE Transactions on Industrial Informatics
12. Jiang W, Li H, Xu G, Wen M, Dong G, Lin X (2019) Ptas: privacy-preserving thin-client authentication scheme in blockchain-based pki. Future Gen Comput Sys 96:185–195
13. Kamara S, Papamanthou C, Roeder T (2011) Cs2: a semantic cryptographic cloud storage system. Technical report, Tech. Rep. MSR-TR-2011-58, Microsoft Technical Report (May 2011) http...
14. Kamara S, Papamanthou C, Roeder T (2012) Dynamic searchable symmetric encryption. In: ACM conference on computer & communications security
15. Li H, Liu D, Dai Y, Luan TH, Yu S (2018) Personalized search over encrypted data with efficient and secure updates in mobile clouds. IEEE Trans Emerging Topics Comput 6(1):97–109
16. Li H, Yang Y, Dai Y, Yu S, Xiang Y Achieving secure and efficient dynamic searchable symmetric encryption over medical cloud data. IEEE Transactions on Cloud Computing, pp 1–1, accepted 2017, to appear. https://doi.org/10.1109/TCC.2017.2769645
17. Liu L, De Vel O, Han Q-L, Zhang J, Xiang Y (2018) Detecting and preventing cyber insider threats: a survey. IEEE Communications Surveys & Tutorials 20(2):1397–1417
18. Merkle RC (1987) A digital signature based on a conventional encryption function. In: Conference on the theory and application of cryptographic techniques. Springer, Berlin, pp 369–378
19. Ning C, Cong W, Ming L, Ren K, Lou W (2011) Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: Infocom. IEEE
20. Ren H, Li H, Dai Y, Yang K, Lin X (2018) Querying in internet of things with privacy preserving: challenges, solutions and opportunities. IEEE Network 32(6):144–151
21. Song DX, Wagner D, Perrig A (2002) Practical techniques for searches on encrypted data. In: IEEE symposium on security & privacy, pp 44–55
22. Sun W, Wang B, Cao N, Li M, Li H (2014) Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. IEEE Trans Parallel Distrib Syst 25(11):3025–3035
23. Wan Z, Deng RH (2016) Vpsearch: achieving verifiability for privacy-preserving multi-keyword search over encrypted cloud data. IEEE Trans Dependable Secure Comput PP(99):1–1
24. Wang C, Cao N, Li J, Ren K, Lou W (2010) Secure ranked keyword search over encrypted cloud data. In: IEEE international conference on distributed computing systems
25. Wang Q, He M, Du M, Chow SSM, Zou Q (2018) Searchable encryption over feature-rich data. IEEE Trans Dependable Secure Comput PP(99):1–1
26. Wang X, Cheng X, Yu X (2019) Efficient verifiable key-aggregate keyword searchable encryption for data sharing in outsourcing storage. IEEE Access 8:11732–11742
27. Wu DN, Gan QQ, Wang X (2018) Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting. IEEE Access 6:42445–42453
28. Xu G, Li H, Dai Y, Yang K, Lin X (2019) Enabling efficient and geometric range query with access control over encrypted spatial data. IEEE Trans Inform Forensics Secur 14(4):870–885
29. Xu G, Li H, Liu S, Wen M, Lu R (2019) Efficient and privacy-preserving truth discovery in mobile crowd sensing systems. IEEE Trans Vehicular Technol 68(4):3854–3865
30. Xu G, Li H, Liu S, Yang K, Lin X (2020) Verifynet: secure and verifiable federated learning. IEEE Trans Inform Forensics Secur 15(1):911–926

31. Xu G, Li H, Ren H, Yang K, Deng RH (2019) Data security issues in deep learning: attacks, countermeasures and opportunities. IEEE Communications Magazine 57(11):116–122
32. Zhang J, Xiang Y, Wang Y, Zhou W, Xiang Y, Guan Y (2012) Network traffic classification using correlation information. IEEE Transactions on Parallel and Distributed systems 24(1):104–117
33. Zhang Q, Fu S, Jia N, Xu M (2018) A verifiable and dynamic multi-keyword ranked search scheme over encrypted cloud data with accuracy improvement. In: International conference on security and privacy in communication systems

Haoyang Wang is a Master's student at the State Key Laboratory of Integrated Service Networks of Xidian University. He received his B.S. degree in software engineering from Northwestern University in 2018. Now he is studying for his M.S. degree in computer science from Xidian University. His research interests are cloud computing security, Searchable Encryption and IoV Security.

Kai Fan received his B.S., M.S. and Ph.D. degrees from Xidian University, P. R. China, in 2002, 2005 and 2007, respectively, in Telecommunication Engineering, Cryptography and Telecommunication and Information System. He is working as an associate professor in State Key Laboratory of Integrated Service Networks at Xidian University. He published over 70 papers in journals and conferences. He received 9 Chinese patents. He has managed 5 national research projects. His research interests include IoT security and information security.

Hui Li was born in 1968 in Shaanxi Province of China. In 1990, he received his B. S. degree in radio electronics from Fudan University. In 1993, and 1998, he received his M. S. degree and Ph.D. degree in telecommunications and information system from Xidian University respectively. He is now a professor of Xidian University. His research interests include network and information security.

Yintang Yang was born in 1962 in Hebei Province of China. He received his Ph.D. degree in semi conductor from Xidian University. He is now a professor at Key Lab. of Minist. of Educ. for Wide Band-Gap Semicon. Materials and Devices of Xidian University, Xi'an China. His research interests include semiconductor materials and devices, network and information security.