Towards Efficient Verifiable Multi-Keyword Search over Encrypted Data based on Blockchain

Wan Shan Xu[1,2], Jian Biao Zhang[1,2], Yi Lin Yuan[1,2], Xiao Wang[3], Yan Hui Liu[1,2], Muhammad Irfan Khalid[1,2]

[1] Faculty of Information Technology, Beijing University of Technology, Beijing, China
[2] Beijing Key Laboratory of Trusted Computing, Beijing, China
[3] Department of Information Science and Technology, Tianjin University of Finance and Economics, Tianjin, China

Corresponding Author:
Jian Biao Zhang[1,2]
100 Pingyuan Park, Beijing, 100124, China
Email address: zjb@bjut.edu.cn

Abstract

Searchable symmetric encryption (SSE) provides an effective way to search encrypted data stored on untrusted servers. As we all known, the server is not trusted, so it is indispensable to verify the results returned by it. However, the existing SSE schemes either lack fairness in the verification of search results, or do not support the verification of multiple keywords. To address this, we design a multi-keyword verifiable searchable symmetric encryption scheme based on blockchain, which provides an efficient multi-keyword search and fair verification of search results. We utilize bitmap to build search index in order to improve search efficiency, and use blockchain to ensure fair verification of search results. The bitmap and hash function are combined to realize lightweight multi-keyword search result verification, compared with the existing verification schemes using public key cryptography primitives, our scheme reduces the verification time and improves the verification efficiency. In addition, our scheme supports the dynamic update of files and realizes the forward security in update. Finally, formal security analysis proves that our scheme is secure against Chosen-Keyword Attacks (CKA), experimental analysis demonstrations that our scheme is efficient and viable in practice.

Introduction

With the development of artificial intelligence, Internet of things, Internet of vehicles and other emerging technologies, more and more enterprises and individuals outsource local data to the cloud, thereby reducing storage and management overhead. However, security and privacy concerns still hinder the deployment of cloud storage system. Although data encryption can eradicate such concerns to some extent, it becomes difficultfor users to search over the data.

Searchable symmetric encryption (SSE) provides an efficient mechanism to solve this, which enables users to search encrypted data efficiently without decryption. Since SSE was first proposed by Song (Song, Wagner & Perrig, 2000), how to perform efficient and versatile search on encrypted data has always been an important research direction. The existing SSE schemes mainly use linked lists and vectors to build indexes, the cloud server needs to traverse the whole list or vector to search for matching results during a query, which incurs high search overhead. In addition to efficient search, dynamic updates are also very important in SSE. (Zhang, Katz & Papamanthou, 2016) has shown that adversaries can infer the critical information through the file injection attacks during the dynamic update of the SSE, while the forward-secure SSE can avoid this. Therefore, the forward security of the scheme must be fully considered when designing the SSE scheme.

Verifiability of the search results is another important research issue for SSE. Since The cloud server is untrusted, which may returns incorrect or incomplete results due to system failures or cost savings, so, it is necessary to verify the search results. In 2012, (Qi & Gong, 2012) proposed the concept of verifiable SSE (VSSE) and constructed a verifiable SSE scheme based on word tree. Following this work, a great many VSSE schemes are proposed (Kurosawa & Ohtaki, 2012; Zhu, Liu & Wang, 2016; Liu et.al 2017; Zhang et.al 2019;Chen et.al 2021). In these schemes, the verification is mainly performed by users, but the user may forge verification results to save costs, so the reliability of the verification cannot be guaranteed. To address this, some researchers(Hu et.al 2018; Li et.al 2019; Guo, Zhang & Jia , 2020) introduce blockchain into SSE to verify search results, which guarantees the fairness and reliability of the verification. Although blockchain achieves fair verification of search results, but the existing schemes are only for a single keyword, and there is little research on fair verification for multi-keywords.

In this paper, we introduce a verifiable multi-keyword SSE scheme based on blockchain, which can perform efficient multi-keyword search, ensures the fairness of verification, and supports the dynamic update of files. To our knowledge, this is the first scheme to verify the search results of multi-keywords fairly. In general, the contributions of this paper are summarized as follows:

- Our scheme realizes efficient multi-keyword search and verification of search results, at the same time, our scheme supports dynamic update of files and achieves forward security.
- Our scheme utilizes blockchain to verify the search results, ensuring the reliability and fairness of the verification results. Combining bitmap index and hash function, we realize lightweight multi-keyword verification to improve verification efficiency.
- We formally prove that our scheme is adaptively secure against CKA, and we conduct a series of experiments to evaluate the performance of our scheme.

Related Works

Searchable Symmetric Encryption

Since SSE was proposed, a number of works have been done to improve search efficiency, rich expression and advanced security. The first SSE scheme (Song, Wagner & Perrig, 2000) enables users to search keywords through full-text scanning, search time increases linearly with the size of files, which is impractical and inefficient. To improve efficient, Curtmola et.al (2006) proposed an inverted index SSE, which achieves sub-linear search time, and gives a definition of SSE security, but this scheme does not support dynamic operations. Wang, Cao & Ren (2010) expanded the scheme of Curtmola et.al (2006) to support dynamic operations, and proved that the scheme was adaptively secure against chosen-keyword attacks (CKA2-secure). For the schemes that support dynamic operation, forward security is critically crucial. The research of Cash et.al (2013) and Zhang, Katz & Papamanthou (2016) indicated that in the SSE scheme without forward security, the adversary can recover most of the sensitive information in ciphertext at a small cost, their research shows the importance of forward security.

Multi-keyword search is a crucial means to improve search efficiency. In single-keyword search scheme (Song, Wagner & Perrig,2000; Curtmola et.al, 2006; Wang, Cao & Ren, 2010), the server returns some irrelevant results, while the multi-keyword search (Cash et.al, 2013; Lai et.al, 2018; Xu et.al, 2019;Liang et.al 2020;Liang et.al 2021) gains higher search accuracy and more accurate results. To further improve search efficiency, Abdelraheem et.al (2016) proposed an SSE scheme on encrypted bitmap indexes to support multi-keyword search, but requires two rounds of interactions with the cloud server. Zuo et.al (2019) proposed a secure SSE scheme based on bitmap index which supports dynamic operations with forward and backward security, but this scheme lacks the verification of the results.

Verifiable Searchable Symmetric Encryption

In SSE, it is necessary to verify the results since the server is untrusted. Qi & Gong (2012) proposed the concept of verifiable searchable symmetric encryption (VSSE) and constructed a VSSE scheme based on word tree. Along this direction, some other VSSE schemes (Kurosawa & Ohtaki, 2012; Zhu, Liu & Wang ,2016; Liu et.al ,2017,Miao et.al 2021) are proposed. These schemes are the verification of single keyword search results, Azraoui et.al (2015) combined polynomial-based accumulators and Merkle trees to achieve conjunctive keyword verification. Wan & Deng (2018) used homomorphic MAC to verify the results of multi-keyword search. Li et.al (2021) utilized bitmap index to gain high efficiency of multi-keyword search, and verified the results by RSA accumulator. Ge et.al (2021) and Liu et.al (2021) proposed their verifiable schemes in the Internet of things. These schemes verify the results of multi-keyword search by public key cryptography primitives, which is computationally expensive and inefficient. What is more, these multi-keyword search verifiable schemes mainly focus on verifying the returned files are valid and whether the files really contains the query keywords, but they didn’t ensure all files containing the query keywords are returned.

Verifiable Searchable Symmetric Encryption Based on Blockchain

In the existing SSE schemes, the verification of search results is performed by users. However, users may forge verification results for economic benefits, which damages the fairness of verification. To solve this, a flexible and feasible method is to adopt blockchain to verify search results, which uses the non-repudiable property of the blockchain to ensure the reliability and fairness of verification. Hu et.al (2018) built a distributed, verifiable and fair ciphertext retrieval scheme based on blockchain. Li et.al (2019) proposed a verifiable scheme combined blockchain and SSE, which can verify the results automatically and reduce the calculation of users. Guo, Zhang & Jia (2020) used the blockchain to realize the public authentication of search results, and ensures forward security of dynamic update. Although these schemes realize the fair verification of search results, but they are mainly for single keyword search, whereas there is little research on the fair verification of multi-keyword. Comparison results with existing schemes are shown in Table 1.

Preliminaries

Bitmap

To improve search efficiency, we use the bitmap (Spiegler & Maayan, 1985) to build inverted index. Bitmap uses a binary string to store a set of information, which can effectively save storage space, and it has been widely used in the field of ciphertext retrieval. In our scheme, each keyword w_i corresponds to a bitmap, which contains ℓ bits, ℓ is the number of files in the system, if the $i-$th document contains w_i the value of ℓ in position i is 1, otherwise 0. For example, there are four files ($\mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3, \mathrm{f}_4$) and two keywords ($\mathrm{w}_1, \mathrm{w}_2$), in Fig.1, w_1 is contained in f_1 and f_3, w_2 is contained in f_2 and f_3, the bitmap of w_1 and w_2 are 1010 and 0110. If we want to search files that contains both w_1 and w_2, we need to do AND operation on the two bitmaps, i.e. $1010 \wedge 0110 = 0010$,that indicates that f_3 contains both w_1 and w_2.

Blockchain

Blockchain is a distributed database, which is widely used in emerging cryptocurrencies to store transaction information such as bitcoin. The blockchain has the features of decentralization, transparency and unforgeability. There is no central server in the blockchain, all nodes participate in the operation and generate the calculation results, the information stored on the blockchain can be seen by all nodes in the network. All nodes of the blockchain share the same data record, under the action of the consensus mechanism, a single node cannot modify the data stored on the chain. The above characteristics of blockchain make it suitable to be a trusted third party for fair verification.

Method

System Model

The system model of our scheme is shown in Fig.2, there are four entities in the system: data owner, cloud server, data user, blockchain. For the files $\mathbf{F}$ in the system, data owner extracts all keywords and generates a keyword set $\mathbf{W}$. Data owner encrypts files to a database T, builds an encrypted index $\mathrm{T}_{\mathcal{B}}$ and a checklist B, $\mathrm{T}_{\mathcal{B}}$ and T are sent to cloud server, $\mathrm{T}_{\mathcal{B}}$ and B are sent to blockchain. When a data user joins the system, it sends an authentication request to the data owner, obtains keys and system parameters. During a query, the data user generates search token $TK_{i,Q}$ according to the keywords to be queried with the help of keys and system parameters, and then sends it to cloud server and blockchain, respectively. Cloud server provides storage services for index $\mathrm{T}_{\mathcal{B}}$ and T. In addition, the cloud server performs ciphertext retrieval according to the search token $TK_{i,Q}$, and sends the matched results to blockchain for verification.

To verify the search results of multiple keywords, the blockchain performs two steps: 1) benchmark. On receiving $TK_{i,Q}$, the blockchain performs multi-keyword search on the index $T_{\mathcal{B}}$ to get the identifiers ID of files that meets the query, then gets the corresponding hash values $\mathbb{H}$ of files from the checklist B according ID, and computes the benchmark Acc using $\mathbb{H}$; 2) verification. After receiving the results returned by cloud server, the blockchain computes the hash values $\mathbb{H}'$ of results and computes the verification value Acc', then the blockchain compares Acc and Acc' to generate the proof. The proof and search results are sent to data user, the verification is completed.

Threat Model

Like other verifiable SSE schemes (Soleimanian & Khazaei, 2019), we assume that the cloud server is malicious, which may return an incorrect or incomplete search result for selfish reasons, such as saving bandwidth or storage space. In addition, we assume that the data user is also untrusted, since it may forge the verification results for economic benefits. The data owner and blockchain are trusted, they execute the protocols in the system honestly.

Algorithm Definitions

Our scheme includes eight polynomial time algorithms, $\prod = \{\mathrm{Keygen}, \mathrm{Setup}, \mathrm{ClientAuth}\ \mathrm{TokenGen}, \mathrm{Search}, \mathrm{Verify}, \mathrm{UpdateToken}, \mathrm{Update}\}$, and the details are as follows:

- $K \leftarrow \mathbf{KeyGen}(1^{\lambda})$, takes system parameter λ as input, and outputs system keys K.
- $(\mathrm{T}, \mathrm{T}_{\mathcal{B}}, \mathrm{B}) \leftarrow \mathbf{Setup}(K, \mathbf{W}, \mathbf{F})$, takes system keys K, the keyword set $\mathbf{W}$ and the set of files $\mathbf{F}$ as input, outputs a database of encrypted files T, an encrypted index $\mathrm{T}_{\mathcal{B}}$ and a checklist B.
- $(K_1, \sum) \leftarrow \mathbf{ClientAuth}(\mathbb{A}_i)$, takes the attribute $\mathbb{A}_i$ of user as input, outputs secret key K_1 and the keyword status $\sum$.
- $TK_{i,Q} \leftarrow \mathbf{TokenGen}(K_1, \overline{\mathrm{W}})$, takes secret key K_1, a set of keywords to query $\overline{\mathrm{W}} = \{w_1, w_2, ..., w_t\}$, outputs the search token $\mathrm{TK}_{i,Q}$.
- $(R, Acc) \leftarrow \mathbf{Search}(\mathrm{T}, \mathrm{T}_{\mathcal{B}}, \mathrm{B}, TK_{i,Q})$, takes search token $\mathrm{TK}_{i,Q}$, the encrypted database T, encrypted index $\mathrm{T}_{\mathcal{B}}$ and the checklist B as input , and outputs the search results R and the benchmark Acc.

- $(R, proof) \leftarrow \mathbf{Verify}(R, Acc)$, takes the search results R, and the benchmark Acc as input, outputs the verification proof $proof$ and results R.
- $(\tau_s, \tau_b) \leftarrow \mathbf{UpdateToken}(\overline{\mathbf{F}}, \mathbf{W}', K)$, takes the set of files to update $\overline{\mathbf{F}}$, the set of keywords $\mathbf{W}'$ and system keys $K = \{K_1, K_2, K_3\}$ as input, and outputs the update token (τ_s, τ_b).
- $(\mathrm{T}', \mathrm{T}_{\mathcal{B}}', \mathrm{B}') \leftarrow \mathbf{Update}(\mathrm{T}, \mathrm{T}_{\mathcal{B}}, \mathrm{B}, \tau_s, \tau_b)$, takes encrypted database T, encrypted index $\mathrm{T}_{\mathcal{B}}$ and the update token (τ_s, τ_b) as input, outputs the updated database T', updated index $\mathrm{T}_{\mathcal{B}}'$ and the updated checklist B'.

Security Definitions

We prove the security of our scheme with the random oracle model, which can be executed by two probabilistic games $\mathrm{Real}_{\mathcal{A}}(\lambda)$ and $\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$, and we have the following definitions:

Definition 1 : CKA2-security, for the verifiable multi-keyword search scheme $\prod = \{\mathrm{KeyGen, Setup, ClientAuth, TokenGen, Search, Verify, Update}\}$, let $\mathcal{L} = \{\mathcal{L}_{\mathrm{setup}}, \mathcal{L}_{\mathrm{search}}, \mathcal{L}_{\mathrm{update}}\}$ be the leakage function, $\mathcal{A}$ is the adversary and $\mathcal{S}$ is the simulator, there are two probabilistic experiments:

$\mathrm{Real}_{\mathcal{A}}(\lambda)$ **:** The challenger runs $\mathrm{KeyGen}(1^{\lambda})$ to generate secret key $K = \{K_1, K_2, K_3\}$, the adversary $\mathcal{A}$ outputs $\mathbf{F}$ and $\mathbf{W}$. The challenger triggers this experiment to run $\mathrm{Setup}(K, \mathbf{W}, \mathbf{F})$, outputs the index $\mathrm{T}_{\mathcal{B}}, \mathrm{T}$ and B, which are sent to $\mathcal{A}$. $\mathcal{A}$ generates a series of adaptive queries $Q = \{q_1, q_2, ..., q_t\}$, for each $q_i \in Q$, the challenger generates search or update tokens, $\mathcal{A}$ receives those tokens and generates a bit b as the output of this experiment.

$\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$: The adversary $\mathcal{A}$ outputs $\mathbf{F}$ and $\mathbf{W}$, the simulator $\mathcal{S}$ generates the index $\mathrm{T}_{\mathcal{B}}, \mathrm{T}$ and B through $\mathcal{L}_{\mathrm{Setup}}$, $\mathcal{A}$ receives them. $\mathcal{A}$ generates a series of adaptive queries $Q = \{q_1, q_2, ..., q_t\}$, for each $q_i \in Q$, the simulator $\mathcal{S}$ generates search or update tokens with $\mathcal{L}_{\mathrm{Search}}$ and $\mathcal{L}_{Update}$, $\mathcal{A}$ receives those tokens and generates a bit b as the output of this experiment.

If for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, there exist an efficient simulator $\mathcal{S}$, which satisfies that:

$$|\Pr[\mathrm{Real}_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda) = 1] \leq negl(\lambda)$$

, we say $\prod$ is $\mathcal{L}-$ secure against CKA2, where $negl$ is an negligible function and λ is the security parameter.

Construction

In this section, we present the construction of our scheme in detail. We take bitmap as index structure to achieve efficient search over encrypted data, and use blockchain to verify the search results. The bitmap is utilized to build the inverted index to achieve the optimal search time $\mathcal{O}(|q|)$,where q is the keywords in search and $|q|$ is the number of q.

In our scheme, the blockchain is used to fairly verify the search results. In Setup, the data owner calculates the hash value of files, generates a checklist B and saves it on the blockchain. During the verification, the blockchain smart contract computes the hash values of search results returned by the server and compares them with the existing results to obtain the verification results.

Specifically, in the single keyword setting, the blockchain stores the corresponding benchmark directly since the results corresponding to the keywords are determined. However, it's impossible in multi-keyword search because the search results are variable, which can only store the verification value of each file. To ensure the credibility of the search results, the blockchain also needs to perform multi-keyword search to obtain the search results. Therefore, we save the index $\mathrm{T}_{\mathcal{B}}$ on the blockchain. During a query, the blockchain executes multi-keyword search to get the search results, and read the verification value $hash_i$ of each file in search results to generate the benchmark Acc, then the blockchain compares Acc with search results returned by cloud server to complete the verification.

Proposed Construction

Our scheme contains eight algorithms $\prod=\{\mathrm{KeyGen,Setup,ClientAuth,TokenGen,Search,Verify\ UpdateToken,Update}\}$, let $F:\{0,1\}^* \to \{0,1\}^m$, $H:\{0,1\}^* \to \{0,1\}^n$ be two Pseudo-Random Functions (PRFs), the constructions of our scheme are as follows.

$K \leftarrow \mathbf{KeyGen}(1^\lambda)$: This algorithm is executed by the data owner, given a security parameter $\lambda \in \mathbb{N}$, this algorithm generates the secret key $K=\{K_1,K_2,K_3\}$, where $K_1,K_2,K_3 \xleftarrow{\$} \{0,1\}^\lambda$, K_1,K_2 are used to encrypt the bitmap index for each keyword $\mathrm{w}_i \in \mathbf{W}$, K_3 is used to encrypt files $\mathrm{f}_i \in \mathbf{F}$ and store the hash value of files.

$(\mathrm{T},\mathrm{T}_{\mathcal{B}},\mathrm{B}) \leftarrow \mathbf{Setup}(K,\mathbf{W,F})$: Given a set of files $\mathbf{F}$, a set of keywords $\mathbf{W}$ and the secret keys K, this algorithm builds an encrypted index $\mathrm{T}_{\mathcal{B}}$, a checklist B and a ciphertext database T, as is shown in Algorithm 1. For each file $\mathrm{f}_i \in \mathbf{F}$, id_i is the identifier of f_i, the data owner encrypts f_i by calculating $\mathrm{c}_i \leftarrow \mathrm{Enc}(K_3,\mathrm{f}_i)$, and computes the hash value using $hash_i \leftarrow H(\mathrm{c}_i)$. Then data owner stores c_i and $hash_i$ in $\mathrm{T}[l_i]$ and $\mathrm{B}[l_i]$, respectively.

For each keyword $\mathrm{w}_i \in \mathbf{W}$, data owner generates a bitmap $\mathcal{B}_{w_i}$, if id_j contains keyword w_i, then $\mathcal{B}_{w_i}[m]=1$, where $m=H(id_j \| K_3)$, and the other positions of $\mathcal{B}_{w_i}$ are all 0's. The data owner encrypts $\mathcal{B}_{w_i}$ through $v_{\mathcal{B}} \leftarrow \mathcal{B}_{w_i} \oplus H(t_w \| st_{i+1})$, and store $v_{\mathcal{B}}$ in $\mathrm{T}_{\mathcal{B}}[t_w]$. At the end of the Setup, $(\mathrm{T}_{\mathcal{B}},\mathrm{B})$ and $(\mathrm{T},\mathrm{T}_{\mathcal{B}})$ are sent and stored on blockchain and cloud server, respectively.

$(K_1, \Sigma) \leftarrow \mathbf{ClientAuth}(\mathbb{A}_i)$: It needs to register to the data owner when a new data user who wants to query files on the cloud server joins the system. The data user submits attribute $\mathbb{A}_i$ to the data owner through this algorithm to obtain the keyword status Σ and the key K_1.

$TK_{i,Q} \leftarrow \mathbf{TokenGen}(K_1, \overline{\mathrm{W}})$: It takes the key K_1 and the set of keywords to query $\overline{\mathrm{W}} = \{w_1, w_2, ..., w_t\}$ as input, output a search token $TK_{i,Q}$, as is shown in Algorithm2. For each keyword $w_i \in \overline{\mathrm{W}}$, the data user computes the position l_{w_i} of w_i in index $\mathrm{T}_{\mathcal{B}}$ as $l_{w_i} \leftarrow H(u_{w_i} \| st_i)$, where $u_{w_i} \leftarrow F(K_1, H_1(w_i))$, $st_i \leftarrow \Sigma[w_i]$. Data user sends $TK_{i,Q}$ to cloud server and blockchain, respectively.

$(R, Acc) \leftarrow \mathbf{Search}(\mathrm{T}, \mathrm{T}_{\mathcal{B}}, \mathrm{B}, TK_{i,Q})$: This algorithm takes search token $TK_{i,Q}$, index $\mathrm{T}_{\mathcal{B}}$ and ciphertext database T as input, and outputs search results R. On receiving the search token, the cloud server and blockchain perform the same operations for multi-keyword search. They all parse out the position l_{w_i} of the keyword in the token $TK_{i,Q}$, and get the bitmap $\mathcal{B}_{w_i}$ through $\mathcal{B}_{w_i} \leftarrow v_{\mathcal{B}} \oplus H(K_{w_i} \| l_i)$, $v_{\mathcal{B}} \leftarrow \mathrm{T}_{\mathcal{B}}[l_{w_i}]$. To achieve multi-keyword search, they compute $\mathcal{B} = \mathcal{B}_1 \wedge \mathcal{B}_2 \wedge ... \wedge \mathcal{B}_t$, the cloud server gets files in T according to $\mathcal{B}$ with regard to $\mathcal{B}[i]$=1, and sends them to the blockchain to verify. Similarly, the blockchain gets hash values $\{hash_1, hash_2, ..., hash_s\}$ of files in B according to $\mathcal{B}$, computes $Acc = hash_1 \oplus hash_2 \oplus ... \oplus hash_s$ as the benchmark for verification, and the details are shown in Algorithm 2.

$(R, proof) \leftarrow \mathbf{Verify}(R, Acc)$: This algorithm takes search results R and benchmark Acc as input, outputs search results R and $proof$, and the verify process is shown in Algorithm 3. To verify the integrity of files, the data owner calculates the hash value of each file through $hash_i \leftarrow H(\mathrm{c}_i)$ in the Setup, and adds $hash_i$ to the checklist B, then B is sent to the blockchain. Through algorithm **Search**, the blockchain gets the search result of multiple keywords, obtains the hash value of each file in the result from B, and computes the benchmark Acc .To verify the search results, the blockchain calculates $H_{\overline{W}}$ of R and compares it with Acc .

In Algorithm 3, for all ciphertexts $c_i \in R$, blockchain computes $H_{\overline{W}} \leftarrow H_{\overline{W}} \oplus H(c_i)$, where $H(c_i)$ denotes the hash value of c_i. Blockchain compares $H_{\overline{W}}$ and Acc, if they are equal, the proof is true, otherwise false. At last, the search results R and proof are sent to data user. During the verification, Acc is calculated through the hash value stored on the blockchain, due to the unforgeability of blockchain, thus Acc is unforgeable. In addition, the verification is completed by the blockchain, so the proof is also unforgeable, which ensures the fairness of verification.

$(\tau_s, \tau_b) \leftarrow \mathbf{UpdateToken}(\overline{\mathrm{F}}, \mathrm{W}', K)$: The data owner generates an update token through this algorithm, which takes files $\overline{\mathrm{F}}$, a keyword set W' and secret key K as input, and outputs update token(τ_s, τ_b).For files $\mathrm{f}_k \in \overline{\mathrm{F}}$, the data owner encrypts and calculates the hash value of f_k by $\mathrm{c}_k \leftarrow \mathrm{Enc}(K_3, \mathrm{f}_k)$ and $hash_k \leftarrow H(\mathrm{c}_k)$,respectively. For keywords $\mathrm{W}'=\{w_1, w_2, ..., w_s\}$ that contained in f_k, the data owner generates a bitmap $\mathcal{B}_{w_j}$ for each $w_j \in \mathrm{W}'$, and encrypts $\mathcal{B}_{w_j}$ with $v_{\mathcal{B}} \leftarrow \mathcal{B}_{w_j} \oplus H(l_{w_j} \| st)$, where $l_{w_j} \leftarrow H(u_{w_j} \| st)$, $u_{w_j} \leftarrow F(K_1, H(\mathrm{w_j}))$, $st \leftarrow F(K_2, st_0)$.

$(\mathrm{T}', \mathrm{T}_{\mathcal{B}}', \mathrm{B}') \leftarrow$ **Update**$(\mathrm{T}, \mathrm{T}_{\mathcal{B}}, \mathrm{B}, \tau_s, \tau_b)$: This algorithm takes encrypted database T, index $\mathrm{T}_{\mathcal{B}}$, checklist B, update token (τ_s, τ_b) as input, and outputs updated database T', updated index $T_{\mathcal{B}}'$ and updated checklist B'.The details are shown in Algorithm 4.

Forward security

As described above, dynamic update is the foundation function of an SSE scheme, and forward security is an indispensable component of dynamic update. In Algorithm 4, when updating a file f_i that contains keyword w_j, the data owner retrieves the previous state st_0 from the local state store $\sum$, and generates a new state st through $st \leftarrow F(K_2, st_0)$, where F is a pseudo random function and K_2 is kept in local. To search a keyword w_j, the data user retrieves the current state st_0 from $\sum$,with st_0 data user generates a token to be sent to the cloud server and blockchain. Without the key K_2, the server cannot compute the current state st from a previous state st_0, therefore it cannot get the current token from a previous, considering that the newly added file f_i corresponds to the current token, that means the previous tokens cannot match f_i, then forward security is achieved.

Security Analysis

In this section, we analysis the security of our scheme. For the scheme $\prod = \{\mathrm{KeyGen}, \mathrm{Setup}, \mathrm{ClientAuth}, \mathrm{TokenGen}, \mathrm{Search}, \mathrm{Verify}, \mathrm{UpdateToken}, \mathrm{Update}\}$ with the leakage function $\mathcal{L}=\{\mathcal{L}_{\mathrm{setup}}, \mathcal{L}_{\mathrm{search}}, \mathcal{L}_{\mathrm{update}}\}$, we prove that our scheme is $\mathcal{L}$-secure against CKA2 by proving that $\mathrm{Real}_{\mathcal{A}}(\lambda)$ and $\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$ are computationally indistinguishable.

Theorem 1. Our scheme $\prod$ is $\mathcal{L}$-secure against CKA2, if the encryption algorithm is secure against chosen-plaintext attacks and the pseudo-random function F and H are secure pseudo-random.

Proof: We use a probabilistic polynomial time simulator $\mathcal{S}$ to simulate indexes and a series of tokens. For a PPT adversary $\mathcal{A}$, we prove theorem 1 by the computational indistinguishability between $\mathrm{Real}_{\mathcal{A}}(\lambda)$ and $\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$. In $\mathrm{Real}_{\mathcal{A}}(\lambda)$, $\mathcal{A}$ gets indexes ($\mathrm{T}_{\mathcal{B}}$, T and B), searches token $TK_{i,Q}$ and updates token (τ_s , τ_b) by running Setup , $\mathrm{TokenGen}$ and $\mathrm{UpdateToken}$; in $\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$, $\mathcal{A}$ gets indexes ($\mathrm{T}_{\mathcal{B}}'$, T' and B'), searches token $TK_{i,Q}'$ and updates token (τ_s', τ_b')

by running $\mathcal{L}_{Setup}$, $\mathcal{L}_{Search}$, $\mathcal{L}_{Update}$. We prove that $\mathrm{Real}_{\mathcal{A}}(\lambda)$ and $\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$ are computational indistinguishable by proving that $(\mathrm{T}_{\mathcal{B}}, \mathrm{T}, \mathrm{B}, TK_{i,Q}, \tau_s, \tau_b)$ and $(\mathrm{T}_{\mathcal{B}}', \mathrm{T}', \mathrm{B}', TK_{i,Q}', \tau_s', \tau_b')$ are indistinguishable.

Simulating index. $\mathcal{S}$ initializes three empty tables: $\mathrm{T}', \mathrm{B}', \mathrm{T}_{\mathcal{B}}'$, which are used to store file ciphertexts, verification values and bitmaps, respectively. $\mathcal{S}$ randomly selects a string f_i' of length $|\mathrm{f}_i|$, and encrypts it through $\mathrm{c}_i' \leftarrow \mathrm{Enc}(K_3, \mathrm{f}_i')$, where K_3 is randomly sampled from $\{0,1\}^{\lambda}$. $\mathcal{S}$ maintains three mappings: H, U and L, H stores $(id_i \| K_3, \ell_i')$, U stores $(H(w_i), u_{w_i}')$, and the mapping L stores $(u_{w_i}' \| st_i, t_{w_i}')$. H, U and L are used and updated by the generation of search and update token. $\mathcal{S}$ computes the hash value $hash_i' \leftarrow H(\mathrm{c}_i')$, c_i' is stored in $\mathrm{T}'[l_i']$ and $hash_i'$ is stored in $\mathrm{B}'[l_i']$. $\mathcal{S}$ selects a string $v_{\mathcal{B}}'$ of length $|v_{\mathcal{B}}|$, and stores it in $\mathrm{T}_{\mathcal{B}}'[t_{w_i}] \leftarrow v_{\mathcal{B}}$.

T', B' and $\mathrm{T}_{\mathcal{B}}'$ are simulated by $\mathcal{S}$ through the leakage $\mathcal{L}_{Setup}$, the difference between ($\mathrm{T}_{\mathcal{B}}'$, T', B') and ($\mathrm{T}_{\mathcal{B}}, \mathrm{T}, \mathrm{B}$) is the generation of ($\mathrm{f}_i', \mathrm{c}_i', v_{\mathcal{B}}'$). In ideal environment, ($\mathrm{f}_i', \mathrm{c}_i', v_{\mathcal{B}}'$) are randomly selected, since our encryption algorithm is secure against CKA2 , F and H are secure pseudo-random functions, therefore , the probability that the adversary $\mathcal{A}$ can distinguish between the real environment and the ideal environment is negligible.

Simulating search token. For the keyword w_i to query, $\mathcal{S}$ gets u_{w_i}' from the mapping U through calculating $H(w_i)$, $\mathcal{S}$ checks whether u_{w_i}' is contained in U , if so returns the corresponding entity, otherwise randomly picks a u_{w_i}' in $\{0,1\}^{\ell}$ and stores $(H(w_i), u_{w_i}')$ in U. Similarly, the experiment gets l_{w_i}' from L by $\mathrm{L}[u_{w_i}' \| st_i]$, the search token $TK_{i,Q}' = \{l_{w_i}'\}$. Under the assumption that F and H are secure pseudo-random functions, the adversary $\mathcal{A}$ cannot distinguish $TK_{i,Q}$ and $TK_{i,Q}'$.

Simulating update token. For file f_k to be added, $\mathcal{S}$ first randomly selects a bit string c_k' of length $|\mathrm{f}_k|$, and encrypts it through $\mathrm{c}_k' \leftarrow \mathrm{Enc}(K_3, \mathrm{f}_k')$. $\mathcal{S}$ computes the hash value $hash_k' \leftarrow H(\mathrm{c}_k')$, c_k' is stored in $\mathrm{T}'[l_k']$ and $hash_k'$ is stored in $\mathrm{B}'[l_k']$, where l_k' is obtained from the mapping H . $\mathcal{S}$ maintains a mapping E , which stores (st_0, st), if there is no corresponding entity for st, it randomly picks a st in $\{0,1\}^{l}$, otherwise it returns the corresponding entity. $\mathcal{S}$ gets u_{w_i}' and l_{w_i}' as in search token, selects a string $v_{\mathcal{B}_j}'$ of length $|v_{\mathcal{B}_j}|$, and stores it in $\mathrm{T}_{\mathcal{B}}'[l_{w_j}'] \leftarrow v_{\mathcal{B}_j}'$. The update token ($\tau_s' = \{(l_k', \mathrm{c}_k'), (l_{w_j}', v_{\mathcal{B}_j}')\}$, $\tau_b' = \{(l_k', hash_k'), (l_{w_j}', v_{\mathcal{B}_j}')\}$) and ($\tau_s = \{(l_k, \mathrm{c}_k), (l_{w_j}, v_{\mathcal{B}_j})\}$, $\tau_b = \{(l_k, hash_k), (l_{w_j}, v_{\mathcal{B}_j})\}$) are indistinguishable for the adversary $\mathcal{A}$.

In such a way, $(\mathrm{T}_{\mathcal{B}}, \mathrm{T}, \mathrm{B}, TK_{i,Q}, \tau_s, \tau_b)$ and $(\mathrm{T}_{\mathcal{B}}', \mathrm{T}', \mathrm{B}', TK_{i,Q}', \tau_s', \tau_b')$ are indistinguishable for $\mathcal{A}$, and it means for a PPT adversary $\mathcal{A}$, the probability of distinguishing between $\mathrm{Real}_{\mathcal{A}}(\lambda)$ and $\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$ is negligible, so we have: $|\Pr[\mathrm{Real}_{\mathcal{A}}(\lambda)=1]-\Pr[\mathrm{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)=1] \le negl(\lambda)$. Therefore, our scheme satisfies CKA2-security.

Performance Evaluation

In this section, we evaluate the performance of our scheme by constructing a series of experiments, and compare the experimental results with Li et.al (2021) and Guo, Zhang & Jia (2020). Since Guo, Zhang & Jia (2020) does not support multi-keyword search over encrypted data, we compare our scheme with Li et.al (2021) which supports multi-keyword search. Besides, we compare our scheme with Guo, Zhang & Jia (2020) in terms of dynamic operations.

We deploy our experiments on a local machine with an Intel Core i7-8550U CPU of 1.80GHz、8GB RAM. We use HMAC-SHA-256 for the pseudo-random functions F, SHA-256 for the hash function H. We use AES as the encryption algorithm to encrypt files. We implement the algorithms in data owner, data user and server using Python and construct the smart contract using Solidity, and the smart contract is tested in with the Ethereum blockchain using a local simulated network TestRPC.

For the dataset, we adopt a real-world dataset, Enron email dataset (William, 2015), which contains more than 517 thousand documents. We utilize the Porter Stemmer to extract more than 1.67 million keywords and filter that meaningless keywords, such as “of”, “the”. At last, we build an inverted index with those keywords to improve the search efficiency of the experiment.

Evaluation of Setup

In setup phase, data owner encrypts the files, calculates the initial verification values of ciphertexts, generates the bitmap indexes of keywords, stores them in T, B and $\mathrm{T}_{\mathcal{B}}$, respectively.

First, we compare the setup time of our scheme with Li et.al (2021) and Guo, Zhang & Jia (2020), the setup time is related to the number of files in the index and the number of keywords included in each file. Figure 3 shows the setup time with different number of keywords in each file while the number of files is fixed at 3137, Fig.4 shows the setup time with different number of files when the number of keywords in each file is fixed at 20. Both figures show that the setup time is affected by the number of keywords in each file and the number of files, and the setup time increases linearly concerning the number of keywords and files.

Furthermore, Fig.3 and Fig.4 illustrate that our scheme is more efficient than Li et.al (2021) and Guo, Zhang & Jia (2020) under the same condition in setup time. Since Guo, Zhang & Jia (2020) utilizes the linked list instead of bitmap to build the index, it requires more time than the other schemes. Our scheme takes less time than Li et.al (2021), the reason is that Li et.al (2021) adopts RSA accumulator based on public key encryption to verify multi-keyword search results, in contrast, our scheme utilizes hash functions to verify search results, which reduces the computational overhead greatly.

Evaluation of Search

For the performance of our scheme, we compare the search time of our scheme with Li et.al (2021). Moreover, to better evaluate the performance of the scheme in multi-keyword search, we perform two settings in a query: 5 keywords and 10 keywords, respectively. In figures, the suffix of the icon indicates the number of keywords in a query, i.e., Our scheme_5 indicates the search time spent in our scheme during a query which contains 5 keywords, Our scheme_10 indicates the search time spent in our scheme during a query which contains 10 keywords, similarly, Li et.al (2021)_5 and Li et.al (2021)_10 indicates the search time spent in Li et.al (2021) during a query which contains 5 keywords and 10 keywords, respectively.

Figure 5 shows the search time with different number of keywords in each file when the number of files is fixed at 3137, and Fig.6 shows the search time with different number of files when the number of keywords in each file is fixed at 20. Both figures show that the search time is affected by the number of keywords in each file and the number of files, and the search time increases sub-linearly with the number of keywords and files.

From Fig. 5 and Fig. 6, we can see that the more keywords included in a query, the more time it takes, this is because the more keywords, the search algorithm spends more time to calculate matched files. Another conclusion can be drawn that our scheme is more efficient than Li et.al (2021) in search, the reason is that the same as the setup algorithm, Li et.al (2021) takes more time to calculate the verification values.

Evaluation of Verify

Here, we evaluate the performance of our scheme in verification, we verify the results of searching for 5 keywords and 10 keywords respectively, and compares the verification time with Li et.al (2021),the comparison results are shown in Fig.7 and Fig.8. Figure 7 shows the verification time with different number of keywords in each file when the number of files is fixed at 3137, and Fig.8 shows the verification time with different number of files when the number of keywords in each file is fixed at 20. From those two figures, we can see that the verification time is affected by the number of keywords in each file and the number of files, the verification time increases with the number of keyword and files.

Both figures shows that our scheme gains a higher verification efficiency than Li et.al (2021), the reason is that Li et.al (2021) takes additional time to compute $\mathcal{B}_{\mathrm{f}_i} = y_i \oplus u_i$,where $u_i = F(K_{\mathrm{f}_i} \| r_i), K_{\mathrm{f}_i} = G(K_3, \mathrm{f}_i)$. In addition, the initial verification values in Li et.al (2021) are stored in untrusted server and the verification is performed by the data user, both the server and the user may forge the verification results, while in our scheme, the values are stored in blockchain and the verification is performed by blockchain, cannot be tampered with, hence, our scheme is more fair and secure in verification.

Evaluation of Update

Dynamic update is the important function in SSE, so we evaluate the performance of our scheme in dynamic update by adding a file containing multiple keywords. Figure 9 and Fig.10 show the performance of our scheme, Li et.al (2021) and Guo, Zhang & Jia (2020) in update time, _5 and _10 indicate that the update document contains 5 keywords and 10 keywords, respectively. We observe that the update time increases with the number of files, since the more files, the longer of the bitmap corresponding to a keyword, then the update algorithm performs more operations when calculating $v_{\mathcal{B}} \leftarrow \mathcal{B}_{w_j} \oplus H(u_{w_j} \| st)$. Moreover, the update time is related to the number of keywords contained in the update file, since the more keywords the file contains, the more indexes to update.

Conclusions

In this paper, we present an efficient verifiable multi-keyword search SSE scheme based on blockchain, which accomplishes efficient multi-keyword search and verification. In our scheme, the yardstick of the file is stored on the blockchain, and the verification of the search results is also completed by the blockchain, thus the fairness and reliability of the verification can be ensured. In addition, our solution supports the dynamic update of files and guarantees forward security during the update. Formal security analysis and experimental results show that our scheme is CKA2-security and efficient. Our scheme can be widely used in cloud storage systems such as data outsourcing, cloud-based IoT (Ge et.al, 2021), medical cloud data (Li et.al, 2020),etc., helping to achieve efficient multi-keyword searches, and ensuring the integrity and credibility of search results.

References

[1] Song DX, Wagner D, Perrig A. 2000. Practical Techniques for Searches on Encrypted Data. In: 2000 IEEE Symposium on Security and Privacy. IEEE, 44–55 DOI: 10.1109/SECPRI.2000.848445.

[2] Curtmola R, Garay J, Kamara S, Ostrovsky R. 2006. Searchable Symmetric Encryption:Improved Definitions and Efficient Constructions. In 13th ACM Conference on Computer and Communications Security. ACM, 79–88.

[3] Wang C, Cao N, Li J, Ren K. 2010. Secure Ranked Keyword Search over Encrypted Cloud Data. in IEEE 30th international conference on distributed computing systems. IEEE, 253-262 DOI: 10.1109/ICDCS.2010.34.

[4] Cash D, Jarecki S, Jutla CS, Krawczyk H, Rosu M, Steiner M. 2013. Highly-scalable Searchable Symmetric Encryption with Support for Boolean Queries. In Advances in Cryptology-CRYPTO, 353–373.

[5] Lai S, Patranabis S, Sakzad A, Liu J, Mukhopadhyay D, Steinfeld R, Sun S, Liu D, Zuo C. 2018. Result pattern hidingsearchable encryption for conjunctive queries. Proceedings of the

2018 ACM SIGSAC Conference on Computer and Communications Security. ACM, 745–762 DOI: 10.1145/3243734.3243753.

[6] Xu G, Li HW, Dai YS, Yang K, Lin XD. 2019. Enabling Efficient and Geometric Range Query with Access Control over Encrypted Spatial Data. IEEE Transactions on Information Forensics and Security,14(4): 870-885 DOI: 10.1109/TIFS.2018.2868162.

[7] Qi C, Gong G. 2012. Verifiable Symmetric Searchable Encryption for Semi-Honest-but-Curious Cloud Servers. In: Proc of the IEEE International Conference on Communications. IEEE, 917-922 DOI: 10.1109/ICC.2012.6364125.

[8] Kurosawa K, Ohtaki Y. 2012. UC-Secure Searchable Symmetric Encryption. In: Proc of the International Conference on Financial Cryptography and Data Security. Springer, 285-298 DOI: 10.1007/978-3-642-32946-3_21.

[9] Zhu X, Liu Q, Wang G. 2016. A Novel Verifiable and Dynamic Fuzzy Keyword Search Scheme over Encrypted Data in Cloud Computing. In: Proc of the IEEE Trustcom/BigDataSE/I SPA. IEEE, 845-851 DOI: 10.1109/TrustCom.2016.0147.

[10] Liu Q, Nie XH, Liu XH, Peng T, Wu J. 2017. Verifiable Ranked Search over dynamic encrypted data in cloud computing. In: Proc of the 2017 IEEE/ACM 25th International Symposium on Quality of Service. IEEE, DOI: 10.1109/IWQoS.2017.7969156.

[11] Zhang Z, Wang J, Wang Y. 2019. Towards Efficient Verifiable Forward Secure Searchable Symmetric Encryption. In: Proc of the European Symposium on Research in Computer Security. LNCS, 11736: 304-321 DOI: 10.1007/978-3-030-29962-0_15.

[12] Ge X, Yu J, Zhang H, Hu CY, Li ZP, Qin Z, Hao R. 2019. Towards Achieving Keyword Search over Dynamic Encrypted Cloud Data with Symmetric-Key based Verification. IEEE, 18:490-504 DOI: 10.1109/TDSC.2019.2896258.

[13] AZRAOUI M, ELKHIYAOUI K, ÖNEN M. 2015. Publicly Verifiable Conjunctive Keyword Search in Outsourced Databases. IEEE Conference on Communications and Network Security, Florence. IEEE, 619–627 DOI:10.1109/CNS.2015.7346876

[14] Wang J, Chen X, Sun SF, Liu JK, Man HA, Zhan ZH. 2018. Towards Efficient Verifiable Conjunctive Keyword Search for Large Encrypted Database. In: Proc of the European Symposium on Research in Computer Security. Springer, 83-100 DOI: 10.1007/978-3-319-98989-1_5

[15] Sun W, Liu X, Lou W, Hou YT, Li H. 2015. Catch You if You Lie to Me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data. In: Proc of the IEEE Conf. on Computer Communications. IEEE, 2110-2118 DOI: 10.1109/INFOCOM.2015.7218596

[16] Wan ZZ, Deng RH. 2018. VPSearch: Achieving Verifiability for Privacy-Preserving Multi-Keyword Search over Encrypted Cloud Data. IEEE Transactions on Dependable and Secure Computing, 15:1083-1095 DOI: 10.1109/TDSC.2016.2635128.

[17] Li F, Ma JF, Miao YB, Jiang Q, Liu XM, Kim-Kwang RC. 2021. Verifiable and Dynamic Multi-keyword Search over Encrypted Cloud Data Using Bitmap. IEEE Transactions on Cloud Computing,1-14 DOI: 10.1109/TCC.2021.3093304.

[18] Hu S, Cai C, Wang Q, Wang C, Luo X, Ren K.2018. Searching an Encrypted Cloud Meets Blockchain: A Decentralized, Reliable and Fair Realization. In: Proc of the IEEE Conference on Computer Communications. IEEE, 792-800 DOI: 10.1109/INFOCOM.2018.8485890.

[19] Cai C, Weng J, Yuan X, Wang C. 2021. Enabling Reliable Keyword Search in Encrypted Decentralized Storage with Fairness. IEEE Trans on Dependable and Secure Computing. 18(1):131-144 DOI: 10.1109/TDSC.2018.2877332.

[20] Li HG, Tian HB, Zhang FG, He JJ. 2019. Blockchain-based Searchable Symmetric Encryption Scheme. Computers & Electrical Engineering.73:32-45 DOI: 10.1016/j.compeleceng.2018.10.015

[21] Guo Y, Zhang C, Jia X. 2020. Verifiable and Forward-secure Encrypted Search Using Blockchain Techniques. IEEE International Conference on Communications. IEEE, DOI: 10.1109/ICC40277.2020.9148612.

[22] Spiegler I, Maayan R. 1985. Storage and Retrieval Considerations of Binary Data Bases. Information processing & management, 21(3): 233–254 DOI: 10.1016/0306-4573(85)90108-6.

[23] Soleimanian A, Khazaei S. 2019. Publicly Verifiable Searchable Symmetric Encryption Based on Efficient Cryptographic Components. Designs, Codes and Cryptography, 87(1): 123–147 DOI: 10.1007/s10623-018-0489-y.

[24] Kamara S, Papamanthou C, Roeder T. 2012. Dynamic Searchable Symmetric Encryption. in Proc. ACM Conference on Computer and Communications Security. ACM, 965–976 DOI: 10.1145/2382196.2382298.

[25] Cash D, Grubbs P, Perry J, Ristenpart T. 2015. Leakage-abuse Attacks Against Searchable Encryption. In: Proc. of the 22nd ACM SIGSAC Conference on Computer and Communications Security. ACM, 668–679 DOI: 10.1145/2810103.2813700.

[26] Zhang YP, Katz J, Papamanthou C. 2016. All Your Queries are Belong to Us: the power of file-injection attacks on searchable encryption. In: Proc. of the 25th USENIX Conf. on Security Symposium. Austin, 707–720.

[27] Abdelraheem MA, Gehrmann C, Lindstrom M, Nordahl C. 2016. Executing Boolean Queries on An Encrypted Bitmap Index. In Proc. ACM on Cloud Computing Security Workshop. ACM, 11–22 DOI: 10.1145/2996429.2996436.

[28] Zuo C, Sun S, Liu JK, Shao J, Pieprzyk J. 2019. Dynamic Searchable Symmetric Encryption with Forward and Stronger Backward Privacy. In Proc. European Symposium on Research in Computer Security. Springer, 283–303 DOI: 10.1007/978-3-030-29962-0_14.

[29] William WC. 2015. Enron email dataset. Available at http://www.cs.cmu.edu/~enron/.

[30] Maryam H, Parvaneh A,Hamid HSJ. 2021. Dynamic Secure Multi-keyword Ranked Search over Encrypted Cloud Data. Journal of Information Security and Applications. 61:1-12 DOI: 10.1016/j.jisa.2021.102902

[31] Miao Y, Deng RH, Choo KKR, Liu X,Ning J, Li H. 2021. Optimized Verifiable Fine-grained Keyword Search in Dynamic Multi-Owner Settings. IEEE Transactions on Dependable and Secure Computing. 18(4):1804-1820 DOI: 10.1109/TDSC.2019.2940573.

[32] Chen CM, Tie Z, Wang EK, Khan MK, Kumar S, Kumari S. 2021. Verifiable Dynamic Ranked Search with Forward Privacy over Encrypted Cloud Data. Peer-to-Peer Networking and Applications, 14:2977–2991 DOI: /10.1007/s12083-021-01132-3.

[33] Ge XR, Yu J, Chen F, Kong F, Wang H. 2021. Towards Verifiable Phrase Search over Encrypted Cloud-based IoT Data. IEEE Internet of Things Journal.8(16):12902 – 12918 DOI: 10.1109/JIOT.2021.3063855.

[34] Liu X,Yang X, Luo Y, Zhang Q. 2021. Verifiable Multi-keyword Search Encryption Scheme with Anonymous Key Generation for Medical Internet of Things. IEEE Internet of Things Journal. DOI: 10.1109/JIOT.2021.3056116.

[35] Liang YR, Li YP, Cao Q, Ren F. VPAMS: Verifiable and Practical Attribute-based Multi-keyword Search Over Encrypted Cloud Data. Journal of Systems Architecture. DOI: 10.1016/j.sysarc.2020.101741.

[36] Liang Y, Li Y, Zhang K, Ma L. 2021. DMSE: Dynamic Multi-keyword Search Encryption Based on Inverted Index. Journal of Systems Architecture. DOI: 10.1016/j.sysarc.2021.102255.

[37] Li H, Yang Y, Dai Y, Yu S, Xiang Y. 2020. Achieving Secure and Efficient Dynamic Searchable Symmetric Encryption over Medical Cloud Data. IEEE Transactions on Cloud Computing. 8(2):484-494 DOI: 10.1109/TCC.2017.2769645.