A verifiable ranked ciphertext retrieval scheme based on bilinear mapping

Baohua Huang | Pirong Huang | Hong Yuan | Sheng Liang

School of Computer and Electronic Information, Guangxi University, Nanning, China

Correspondence
Baohua Huang, School of Computer and Electronic Information, Guangxi University, 100 Daxue Road, Nanning 530004, Guangxi, China.
Email: bhhuang66@gxu.edu.cn

Summary
It is prevalent nowadays for data owners to outsource their data to the cloud. Most of the traditional searchable schemes are proposed under the honest and curious model, lacking verification of integrity and correctness of the retrieval results. Because the cloud server is not completely trustworthy and there may be improper execution of the users’ retrieval requests, it is necessary to verify the retrieval results returned by the cloud server. In order to address these issues, we propose a verifiable ranked ciphertext retrieval scheme (VRCRS) based on bilinear mapping. For the purpose of ranking the keywords and improving the security of ciphertext retrieval scheme, we utilize the lucene search engine toolkit to score keywords, and improve the traditional inverted index structure for building a secure inverted index structure. By adding dummy words into the dictionary of keywords, the proposed scheme can resist the keywords statistics attack of malicious server. To ensure the correctness and integrity of returned ciphertext, we apply bilinear mapping to generate validation tags for keywords. The theoretical analysis and experimental results show that the proposed scheme is secure and efficient. VRCRS can actually identify the illegal behavior of cloud server such as tampering and forgery.

KEYWORDS
bilinear, ciphertext retrieval, cloud storage, integrity verification, inverted index

1 INTRODUCTION

With the rapid development of computer technology and network application, the data access demand and data storage capacity requirement of users are increasing. More and more users choose to store their data in the cloud to save local physical storage resources and simplify data management. However, when users outsource data to the cloud storage server, it will inevitably bring data privacy and security problems. In order to ensure the confidentiality of the data on cloud servers, the usual way is to encrypt the data and store it in the form of ciphertext on cloud servers, which will limit the accessibility of outsourced data. The encrypted data make the traditional plaintext retrieval scheme no longer feasible. Therefore, how to retrieve the ciphertext data has become an urgent problem. In order to solve the above problems, Song et al. first proposed a linear scanning algorithm in 2000. Because of scanning the full text, this scheme has low retrieval efficiency and is vulnerable to statistical attacks. Its search overhead will increase linearly with the increase of database size, which makes it inefficient in practical applications. Since then, many researchers have focused on secure keyword search through encrypted data. Kamara and Papamanthou proposed a ciphertext retrieval scheme to support dynamic updating of data. By expanding the inverted index, a keyword search dictionary was designed to achieve sublinear search time. Curmola...
et al. designed symmetric searchable encryption schemes SSE-1 and SSE-2 with high security and efficiency under the adaptive and nonadaptive model, respectively. However, the above schemes are all based on the model of “honesty and curiosity” of cloud server. In real environment, cloud server may be malicious, delete files that users do not use for a long time to save storage space. The problems of storage hardware and software may lead to damage of outsourced files, so the dishonest cloud server may fake search results to deceive users. In order to ensure the correctness and integrity of the data, it is necessary to verify the search results returned by the cloud server.

Based on Boneh-Lynn-Shacham (BLS) signature, this article designs a verifiable ciphertext retrieval scheme, constructs a secure inverted index structure to achieve sublinear search, and resists keyword attack by inserting keywords into dictionary. Our major contributions are as follows:

1. We improve the traditional inverted index structure to construct a secure inverted index, and we use the score information to rank the search results based on lucene search engine toolkit. The confused keywords are inserted into the dictionary to prevent the cloud server from attacking the keyword frequency according to the search frequency of keywords.
2. We use bilinear mapping to generate validation tags for keywords to verify the correctness and reliability of retrieval results.
3. We verify the feasibility and efficiency of the scheme on real data set.

The remainder of this article is organized as follows. Section 2 describe the related works. In Section 3, preliminaries and security model are formalized. In Section 4 we propose a concrete verifiable ranked ciphertext retrieval scheme (VRCRS). The security analysis and performance evaluation of our scheme are presented in Sections 5 and 6, respectively. Finally, we conclude this article in Section 7.

# RELATED WORK

## 2.1 Ranked searchable encryption

Because the relevance of the queried keyword with each document is different, it will cause unnecessary calculation cost if the cloud server returns all of the matched documents. Sometimes users want to fetch the most relevant documents. Ranked search greatly enhances system usability and file retrieval accuracy by enabling search result relevance ranking instead of sending undifferentiated results to the users. In order to address this issue, Wang et al. proposed a secure and efficient ranked keyword search over outsourced cloud data. Their scheme used TF-IDF to calculate the relevance score. Sun et al. proposed a privacy-preserving multikeyword text search scheme with similarity-based ranking. Their scheme builds the search index based on term frequency and the vector space model with cosine similarity measure. A tree-based index structure and various adaption methods are proposed to improve the search efficiency. Cao et al. proposed a privacy-preserving multikeyword ranked search over encrypted cloud data. In this scheme, they use the secure KNN scheme to capture the relevance of data documents to the search query based on inner product value. Jiang et al. proposed a verifiable multikeyword ranked searchable encryption scheme, which adopts a special data structure called QSet to achieve efficient search efficiency. This special data structure can mask the correspondence between the keyword and the document set that contains the keyword. However, this scheme requires that only the user’s input is completely correct can it work properly. TF-IDF values are widely used to find the relevance scores.

## 2.2 Verifiable searchable encryption

The search results returned by the cloud server may be incomplete or contain some errors. This may be caused by malicious cloud server to save computing resources or due to software/hardware malfunction. To solve this problem, Chai and Gong proposed a verifiable keyword ciphertext retrieval scheme for the first time. In this scheme, cloud servers need to return evidence to prove the correctness of the results.

Liu et al. proposed a verifiable searchable encryption with aggregate keys for data sharing system. This scheme enables each authorized user to confidentially retrieve encrypted documents selectively shared by a document provider using a single aggregate key, and to verify the results using the same key. This scheme chooses the Bloom filter as a verification tool, but it does not support results ranking.

Ji et al. proposed a searchable encryption-enabling verifiable fuzzy keyword search over encrypted data in cloud computing. This scheme generates an authentication label for each keyword to detect the malicious behavior of the cloud server. It adopts the linked list as the index structure to build the index efficiently. However, this scheme does not support multikeywords searching and does not support aggregate authentication key. And its every keyword matches an authentication label, when the user initiates multiple rounds of query request, it will return multiple labels. These labels can cause additional communication burden.

Ge et al. proposed a conjunctive keyword searchable encryption scheme with an authentication mechanism that can efficiently verify the integrity of search results. This scheme is based on the dynamic searchable symmetric encryption and adopts the Merkle tree and bilinear map accumulator to prove the correctness of set operation. However, this scheme requires the cloud server to return a complete keyword verification path, which leads to a larger communication bandwidth. It also does not support top-k search.
Wang et al. proposed a verifiable conjunctive keyword search scheme by leveraging accumulator. This scheme can ensure correctness and completeness of search result even if an empty set is returned. The search cost of this scheme depends on the number of documents matching with the least frequent keyword, which is independent of the entire database. In addition, the scheme introduces a sample check method to check the completeness of search result, which can achieve low false-positive by checking only a fraction of the search result. However, this scheme needs to perform nonmembership verification for each document individually.

In addition, Sun et al. proposed a verifiable keyword combination query scheme, which uses bilinear mapping cumulative tree as its validated data structure. Wang et al. proposed a scheme to verify the correctness of search results by using hash linked list. Zhu et al. and Liu et al. proposed a dynamic verifiable ciphertext query scheme, which uses RSA accumulator based on public key cryptosystem to verify the correctness of search results, but the search efficiency is relatively low. Azraoui et al. uses Merkle hash tree to verify the integrity of retrieval results, but this scheme requires cloud servers to return a complete verification path, which requires a large communication overhead. Du et al. uses bilinear mapping to verify the retrieval results, but its proof tag generation algorithm involves the calculation of multiple matrices and bilinear pairings, which makes the tag generation less efficient. And Miao et al. proposed a verifiable conjunctive keywords search over encrypted data without secure channel. In addition, a number of verifiable searchable encryption schemes are based on the cryptographic signature schemes.

3 | NOTATIONS AND PRELIMINARY

We give some definitions and review related cryptographic knowledge about bilinear pairing, complexity assumption, the system architecture, and the security model.

3.1 | Notations

Table 1 lists some notations utilized in this article.

3.2 | Bilinear maps

Let \( G \) and \( G_1 \) be two multiplication cyclic groups of order \( p \). \( Z_p \) is a finite field. Let \( g \) be a generator of \( G \) and \( e: G \times G \rightarrow G_1 \) be a bilinear map with the properties:

- Bilinearity: for all \( u, v \in G \) and \( a, b \in Z_p \), we have \( e(u^a, v^b) = e(u, v)^{ab} \).
- Nondegeneracy: there exists \( u, v \in G \) such that \( e(u, v) \neq 1 \).
- Computability: for all \( u, v \in G \), there is an efficient algorithm to compute \( e(u, v) \).

3.3 | CDH assumption

**CDH problem:** Given a group \( G \) with prime order \( p \), a generator \( g \), and elements \( g^a, g^b \in G \), where \( a \) and \( b \) are chosen randomly from \( Z_p \). The CDH problem is to compute the value \( g^{ab} \).

**CDH assumption:** We say that the CDH assumption holds if all probabilistic polynomial time adversaries have at most negligible advantage in solving the CDH problem.

3.4 | System architecture

System architecture: The model of verifiable ranked ciphertext retrieval system proposed in this article includes three entities: data owner, cloud server, and data user. The system model diagram is shown in Figure 1.

1. **Data owner (DO):** First, DO participles the local plaintext document set, and then encrypts the keywords, inserts obfuscated words into the keyword dictionary when constructing the keyword set. Second, DO uploads the encrypted document and encrypted index to the cloud server. Moreover, DO sends the decryption key and authentication tag to authorized users through the secure channel, and grants data users the right to query and verify the search results.
2) **Cloud server (CS):** Store encrypted data and encrypted index, which are uploaded by data owner, perform retrieval operation on secure inverted index according to query request submitted by data user, return ciphertext document identifier of the top-K documents, which most relevant to the query keywords, and return the validation proof set.

3) **Data user (DU):** The authorized user by the data owner. According to the requirement, the data user sends the trapdoor TD generated from keywords of interest to the CS, and requests the cloud server to perform the retrieval operation then return the previous K documents and their verification evidence. Subsequently, data user uses validation information to perform validation operations. If the validation passes, he or she accept the retrieval results, otherwise he or she refuse to retrieve results. Finally, the encrypted document is decrypted to get the plaintext document containing keywords.
3.5 Definition and security model

Thread model: The same as references, it is assumed that the cloud server is a semihonest and curious adversary, which satisfies following properties:

1. In order to reduce computing overhead or save download bandwidth, or due to malware and software bugs, the cloud server may only return partial results that meet the requirements, and may even falsify or tamper with retrieval results.
2. In order to save storage space, cloud servers may delete some documents or index files.
3. Cloud servers are curious, and may analyze additional information from data stored by users or the query trapdoors, infer or even identify certain keywords.

Definition 1 (correctness). A VRCRS scheme is correct if the following assumption holds: Given \( \forall (PK, SK) \leftarrow \text{Setup}(1^\lambda) \), \( \forall D_i \subseteq D (1 \leq i \leq n) \), \( \forall W \subseteq W \) that \( \text{search}(TD, K, I) \rightarrow (C_{w,K}, \Omega_w) \) \& \( \text{Verify}(PK, SK, C_{w,K}, \Omega_w) \rightarrow (0,1) \) \& \( \text{DecFile}(C_{w,K}, SK) = D_{w,K} \) always returns 1, we can say that the proposed scheme satisfies the correctness. \( D \) is the document set and \( W \) is the keyword set. \( C_{w,K} \) denotes the top-K search results of keyword \( w \) and \( \Omega_w \) denotes its verification tag.

Definition 2 (reliability). Suppose there exists an adversary \( A \). The reliability of VCRS scheme is formally defined as follow: \( \text{Forge}_A^a(\lambda) \) : It is implemented by \( A \). For a query trapdoor, \( A \) forge a false result \( C_{w,K}^* \) and the corresponding forged proof \( \Omega_w^* \) with the condition that \( C_{w,K}^* \neq C_{w,K}, \Omega_w^* \neq \Omega_w \). If the possibility of making algorithm \( \text{Verify}(PK, SK, C_{w,K}^*, \Omega_w^*) = 1 \) establish is negligible, we say that the scheme is reliable. That is \( \Pr[\text{Forge}_A^a(\lambda) = 1] \leq \text{neg} (\lambda) \), which \( \text{neg} (\lambda) \) is a negligible value.

4 CONCRETE SCHEME

In this section, we mainly focus on constructing secure inverted index and generating signatures for keywords.

4.1 Secure inverted index

In this article, inverted index structure is adopted to achieve secure search of ciphertext. On the basis of traditional inverted index, a secure inverted index is designed. Traditional plaintext inverted index structure consists of index file (keywords dictionary) and inverted file (posting list).

As shown in Figure 2, the secure inverted index consists of a search table \( T_s \) and a search array \( A_s \).

The search array \( A_s \) is an array with length \( M = (m + l) \times \#d \), where \( m \) is the number of keywords and \( l \) is the number of the dummy keywords. \( \#d = \max \{ \#w | 1 \leq i \leq m \} \), \( \#w \) denotes the number of documents that contain the keyword \( w \). \( A_s[i] \) represents the value stored in position \( i \) of search array. For each keyword \( w \in W \), the posting list of \( w \) is randomly selected in \( A_s \). \( A_{ni} \) is the posting list of keyword \( w \) with length \( \#d \). \( A_{ci} \) is the posting list of dummy keyword.

Because traditional inverted files retain the location and frequency information of keywords, which are vulnerable to statistical attacks. The main improvements to the traditional plaintext inverted index as follows:

(a) Encrypt each keyword in index file, and replace the plaintext keyword with encrypted keyword. The original logical pointer in the index file is replaced by the encrypted pointer. \( K_{e1}(\cdot) \) and \( P_{e1}(\cdot) \) are pseudorandom functions for encrypting keywords and pointers, respectively.

(b) To resist keyword statistics attacks, obfuscation words are added to index files, and dummy denotes obfuscation words inserted into index files. Since the frequency of keywords is different, the adversary can distinguish different keywords. To address this problem, we can hide the frequency of keyword by padding some random nodes in the posting lists. First, we select the maximal length of posting lists. Then, we pad other dummy nodes in the posting list.

![Figure 2](image-url)
4.2 Dictionary obfuscation keywords

Because the statistical characteristics of the index information and the information of plaintext documents are related closely, attackers may infer the corresponding plaintext by analyzing the index. For each keyword \( w_i \in W \), the number of its occurrences in the document set is \( f_{w_i} \). Assuming that an attacker can obtain a distribution about the frequency of keyword occurrence \( D = (w_i, f_{w_i}) \), the distribution is generally similar to the true distribution of user-stored keywords set.\(^{25}\) Take \( q \) as a keyword of interest to the attacker, the attacker can get the frequency of the word through query repeatedly, and then conduct keyword statistical attacks.

Because high-frequency keywords are vulnerable to statistical analysis attacks, the security inverted index constructed in this article eliminates the frequency difference between high-frequency keywords and low-frequency keywords. As shown in Figure 2, the main methods are as follows:

1. Randomly generate \( l \) confused string and add the \( l \) dummy keywords to the dictionary, which
2. When constructing a posting list of dummy keywords, \( #d \) nodes is selected to each posting list. By the above method, it can eliminate the frequency difference between high-frequency keywords and low-frequency keywords.

4.3 The proposed verifiable searchable scheme

Our verifiable searchable scheme consists of nine algorithms (KeyGen, BuildIndex, EncFile, TrapDoor, AccGen, Search, ProofGen, Verify, DecFile). Now, we describe the concrete construction of our scheme as follows:

\[ \text{KeyGen}(1^\lambda) \rightarrow (PK, SK): \] This algorithm run by DO and takes a security parameter \( \lambda \) as input. It outputs a tuple \( key = (PK, SK) \). The public key is

\[ PK = (e, G, G_1, q, g, v). \]

The private key is \( SK = (p, k_1, k_2, k_3) \). DO chooses two groups \( G \) and \( G_1 \) of order \( q \), a bilinear map \( e: G \times G \rightarrow G_1 \), a generator \( g \) of \( G \), and a hash function \( H_1 : \{0, 1\}^* \rightarrow G \). Meanwhile, DO randomly chooses an element \( p \in Z_q \) as a private key, and computes \( v = g^p \) as a public key. Here, three pseudorandom functions are selected. They are \( K, P, H \), and \( k_1, k_2 \) and \( k_3 \) are their seed keys, respectively. They are defined as follows:

\[ K_{k_1} : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^k \]

\[ P_{k_2} : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^k \]

\[ H_{k_3} : \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^k \]

\[ \text{BuildIndex}(K, D) \rightarrow I: \] DO extract the keywords in \( D \) and set \( W = \{w_1, \ldots, w_m\} \) and for all \( w_i \in W \):

- DO uses pseudorandom function \( K_{k_1}(\cdot) \) to encrypt \( w_i \) and get the encrypted keywords \( C_{w_i} = K_{k_1}(w_i) \).
- Randomly selects \( #w_i \) locations in search array \( A_S \) for creating posting list \( A_W \). The node is \( N_{ij} = \langle id_j, Score, addr(N_{i,j+1}) \rangle \) encrypted with \( H_{k_3}(\cdot) \), that is, \( A_S[addr(N_{i,j})] = (N_{ij} \oplus H_{k_3}(w_i)) \).
- DO randomly chooses a parameter \( u \in G \) and then calculates the tag \( \sigma_i = (H(C_{w_i}) \times u^{\infty})^{\infty} \). And \( H(C_{w_i}) \) indicate the hash value of the encrypted keyword \( C_{w_i} \) to an element on the group \( G \).
- \( T_S \) is a search table, which stores the header pointer of \( A_W \) and the keyword label \( \sigma_i \). The pseudorandom function \( P_{k_2}(\cdot) \) is used to encrypt pointers and keyword labels to obtain \( T_S[K_{k_1}(w_i)] = addr(N_i)||\sigma_i \oplus P_{k_2}(w_i) \). Finally, the secure encrypted index \( I = (T_S, A_S) \) is uploaded to the cloud server.

\[ \text{EncFile}(SK, D) \rightarrow C: \] For each document \( D_i \in D \), the data owner uses AES encryption algorithm to generate the ciphertext document \( C_i \). DO gets the encrypted document set \( C = \{C_1, \ldots, C_n\} \), and uploads \( C \) to the CS.

\[ \text{AccGen}(SK, PK, D, W) \rightarrow \Phi: \] For each keyword \( w_i \), DO calculates the output signature set \( \Phi = \{\Phi_1, \ldots, \Phi_m\} \), where \( \Phi_i = \sigma_i \prod_{j=1}^{m} (id_{j} + q) \), according to the label \( \sigma_i \) of each keyword. The data owner sends the signature set to the data user.
The correctness of the verification of the scheme can be proven through checking Equation (1) is true or not. The left side of equation (1) is:

\[ e(\sigma, g) = e(\prod_{i=1}^{x} \sigma_i^{\omega_i}, g) \]

\[ = e(\prod_{i=1}^{x} h(C\omega_i)^{\omega_i} \times u^*, g) \]

\[ = e(\prod_{i=1}^{x} h(C\omega_i)^{\omega_i} \times u^*, g^3) \]

\[ = e(\prod_{i=1}^{x} h(C\omega_i)^{\omega_i} \times \prod_{i=1}^{x} u^2 C_{\omega_i}^{\omega_i}, v) \]

\[ = e(\prod_{i=1}^{x} h(C\omega_i)^{\omega_i} \times u^{\omega_i} C_{\omega_i}^{\omega_i}, v). \]

The right side:

\[ e(\prod_{i=1}^{x} h(C\omega_i)^{\omega_i} \times u^*, v) \]

\[ = e(\prod_{i=1}^{x}(h(C\omega_i)^{\omega_i} \times u^{\omega_i} C_{\omega_i}^{\omega_i}, v). \]

Because the parameters \( u \) and \( v \) are public, only when the proof \( P \) changes, the two sides of the equation will not be equal. That is to say, the keywords of the search have changed, so we can judge whether the search results contain the encrypted documents of \( W_q \) by checking the equalization of equation (1).

**Theorem 1.** The proposed VRCRS scheme is correct.
In addition, the equation \( e_{i} || \prod_{i=1}^{m} \sigma_{i} \neq 0 \) guarantees the binding and unforgeability of \( \sigma \) and corresponding encrypted index. Finally, the data user decrypts the document to get ranked search results. Therefore, the scheme proposed in this article is correct.

**Theorem 2.** For a semihonest and curious cloud server, the proposed VRCRS scheme is reliable undercomparative Diffie-Hellman (CDH) assumption.

**Proof.** Assuming that the scheme is unreliable and the invalid search results can pass the verification algorithm \( \text{Verify}(C_{w}, P, SK, PK) \); however, due to the unforgeability of the message provided by bilinear mapping, the possibility of forging valid evidence is negligible. If \( A \) can generate a valid forgery, then he can solve the CDH problem in \( G \), which is incompatible for the CDH assumption.

We describe the security game as follow. First, DU sends the challenge message \([K, TD]\) and the CS should return the proof \( P = [\mu, \sigma, \Omega_{\mu}] \) based on the correct encrypted data \( C_{w,K} \). In addition, \( A \) outputs the forgery of verification information based on the corrupted \( C_{w,K}^{*} \), where \( C_{w,K} \neq C_{w,K}^{*} \).

And then, if we let \( \Delta \mu = \mu^{*} - \mu(\Delta \mu \neq 0) \), we can say that \( A \) may successfully win the game if the forged proof information \( P = [\mu^{*}, \sigma^{*}, \Omega_{\mu}^{*}] \) and \( C_{w,K}^{*} \) can pass verification algorithm, otherwise, he fails. If \( A \) wins the security game, then we have \( e(\sigma^{*}, g) = e(\prod_{i=1}^{m} h(C_{w_{i}})^{\nu} \times u^{\nu}, v) \) and \( e(\sigma, g) = e(\prod_{i=1}^{m} h(C_{w_{i}})^{\nu} \times u^{\nu}, v) \), where the later equation is the valid verification information. Therefore, we have \( u^{\nu} = u^{\nu} \).

Given two elements \( \sigma_{1}, \sigma_{2} \in G_{T} \) randomly selected in \( G_{T} \), then exists an element \( x \in Z_{q}^{*} \) to make \( \sigma_{2} = \sigma_{1}^{x} \) because \( G_{T} \) is multiplicative cyclic group. The generator \( g \) can be expressed as \( g = \sigma_{1}^{R_{1}} \sigma_{2}^{R_{2}} \), where \( R_{1}, R_{2} \in Z_{q}^{*} \), then we can have the following equation:

\[
(\sigma_{1}^{R_{1}} \sigma_{2}^{R_{2}})^{x} = 1 \iff \sigma_{1}^{R_{1}x} \sigma_{2}^{R_{2}x} = 1
\]

For DL problem \( (g, g^{x} \in G, \text{output} x \in Z_{q}^{*}) \), given \( \sigma_{1}, \sigma_{2} \), we have \( \sigma_{2} = \sigma_{1}^{x} \), and \( x = \frac{R_{1}x}{R_{2}x} \) provides that \( R_{1} \Delta \mu \neq 0 \). As mention above, \( \Delta \mu \neq 0 \) and \( R_{1} \) is a random element in \( Z_{q}^{*} \), the probability of \( \Delta \mu \neq 0 \) is \( 1 - \frac{1}{q} \). In other words, we can solve the DL problem if \( A \) breaks the security game, which contradicts the DL assumption.

**Theorem 3.** The proposed VRCRS scheme satisfies adaptive chosen keyword attack security.

**Proof.** We introduce leak function and adaptive chosen keyword attack first. \( A \) represents a stateful adversary, \( S \) represents a stateful simulator, and \( L_{1}(\lambda) \) and \( L_{2}(\lambda) \) represent the leak function in the system establishment stage and the search stage, respectively.

The leak function \( L_{1}(\lambda) \) is defined as \( L_{1}(D) = (|D|, \|id(D)|) \| |D| \) denotes the size of each document, \( id(D) \) denotes the document identifier, and \#D represents the size of the document set. The leak function \( L_{2}(\lambda) \) is defined as \( L_{2}(w, TD) = (AP(w), TD) \), \( w \) denotes the keyword of the query, \( D \) is the document set, and \( TD \) is the trapdoor. For two probabilistic experiments \( Real_{\lambda}(\lambda) \) and \( Ideal_{\lambda, S}(\lambda) \) satisfy:

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

If \( \text{negl}(\lambda) \) denotes a negligible function of \( \lambda \), then searchable scheme \( T \) is considered to be secure against adaptive chosen keyword attacks.

\( Real_{\lambda}(\lambda) \): Realized by \( A \), the challenger runs \( \text{KeyGen}(1^{\lambda}) \) to generate the secret key. The adversary \( A \) chooses the document set \( D \) and sends it to the challenger. After that Challenger runs algorithms \( \text{BuildIndex} \) \( SK, D \) \( \rightarrow I \) and \( \text{EncFile} \) \( SK, D \) \( \rightarrow C \), and sends \( I, C \) to \( A \). \( A \) makes a polynomial adaptive query \( q \). For each \( q \), the challenger runs algorithm \( \text{TrapDoor}(w, SK) \) and sends the TD to \( A \). \( \text{BuildIndex} \) is the algorithm of encrypting index, \( \text{EncFile} \) is the algorithm of encrypting document, and \( \text{TrapDoor} \) is the algorithm of generating trapdoor.

\( \text{Ideal}_{\lambda, S}(\lambda) \): It is implemented by \( A \) and \( S \). \( A \) chooses a set of files \( D \) and sends \( I, C \) to \( A \) according to \( L_{1}(\lambda) \). \( A \) executes polynomial adaptive query \( q \). For each \( q \), \( S \) simulates output trapdoor according to leakage function \( L_{2}(D,w) \) and returns it to \( A \).

The proof of Theorem 2 is equivalent to the simulator \( S \), which outputs the existence of polynomial time, so that the outputs of probability experiment \( \text{Real}_{\lambda}(\lambda) \) and \( \text{Ideal}_{\lambda, S}(\lambda) \) are indistinguishable for adversary \( A \), that is,

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

First, simulator \( S \) constructs the security index \( I' = (T_{S}, A_{S}) \), the length of \( A_{S} \) is \( |A_{S}| = \sum_{i=1}^{n} |N_{i}|, |N_{i}| \) denotes the number of documents containing the keyword \( w_{i} \), and \( S \) encrypts \( N_{i} \) with \( c_{i}^{S} \), where \( c_{i}^{S} \) is a string generated by a random function. \( S \) sets \( T_{S} \) as a search table with length \( n \). For \( 1 \leq i \leq n \), \( S \) generates a binary tuple \( (c_{i}^{S}, \text{addr}(A_{S_{i}}) \oplus c_{i}^{S}) \), where \( c_{i}^{S} \) and \( c_{i}^{S} \) are two strings generated by random functions, and \( \text{addr}(A_{S_{i}}) \) is the address of array \( A_{S_{i}} \). If pseudorandom functions \( K, P, H \) satisfy pseudorandomness, then \( A \) cannot distinguish the output of pseudorandom functions from random strings of the same length without knowing the secret key. Therefore, the probability of the adversary distinguishing between real index and simulated index is negligible, that is,

\[
| Pr[\text{BuildIndex}(\text{key}, D, W) \rightarrow I] - Pr[\text{Random} \rightarrow I] | \leq \text{negl}(\lambda)
\]

Similarly, it can be proven that: \( | Pr[\text{EncFile}(\text{key}, D) \rightarrow C] | - | Pr[\text{Random} \rightarrow C] | \leq \text{negl}(\lambda) \) and \( | Pr[\text{KeyGen}(1^{\lambda}) \rightarrow \text{key}] | - | Pr[\text{Random} \rightarrow \text{key}] | \leq \text{negl}(\lambda) \), where \( \text{negl}(\lambda), \text{negl}(\lambda), \) and \( \text{negl}(\lambda) \) represents negligible values, respectively. Since \( A \) tries to win by analyzing the encrypted index, ciphertext, and secret key, then:

\[
| Pr[\text{Real}_{\lambda}(\lambda) = 1] - Pr[\text{Ideal}_{\lambda, S}(\lambda) = 1] | \leq \text{negl}(\lambda) + \text{negl}(\lambda) + \text{negl}(\lambda)
\]
To conclude, the outputs of $\text{Real}_A(\lambda)$ and $\text{Ideal}_A(\lambda)$ for probabilistic polynomial time adversary $A$ is indistinguishable. The searchable scheme proposed in this article satisfies the adaptive chosen keyword attack.

## 6 PERFORMANCE EVALUATION

Table 2 provides a brief comparison of our scheme and some existing schemes. The search algorithm in scheme [21] involves exponential operations and pairing operations in the group $G$ and $G_1$ (given $4P + 2E + 3E1$), while our search algorithm is relevant to the comparative operation only. So our algorithm is more efficient than scheme [21] in search operation.

We also take experiments to evaluate the performance of the proposed scheme. The experimental environment is implemented in Java. The hardware and software configuration are Intel (R) Core (TM) i7-7700HQ (2.80GHz) processor, 16 GB memory, Win10 64-bit operation system, and open source search engine tool Lucene-4.10.3.

In the experiment, a set of 5000 documents is selected as the test data set, and the Lucene word segmentation is used to segment the documents. The stopwords are filtered out. Finally, the keywords in the test data set are extracted to form a 48 000 keywords set.

In the experiment, the indexes of different number of keywords are constructed ($m=2000, 4000, 6000, 8000, 10000$). The encryption algorithm is implemented by JPBC2 library. The elliptic curve uses type A, which corresponds to the symmetric prime order bilinear group. We define several computation operations (e.g., exponential operation $E$ on the multiplicative cyclic group $G$, exponential operation $E1$ on $G_1$, pairing operation $P$). Since the hash operation $H_1$ is more efficient than other operation and we will omit it. We choose Liu’s scheme as experimental comparison, and perform the efficiency comparison in the following aspects: trapdoor generation cost, search cost, proof generation cost, and verification cost. We perform each experiment 10 times to get the average execution time. The results of the experiments are shown in Figure 3.

(a) Index construction time comparison: As we can see from Figure 3A, with the increase of the number of keywords, the time of building plaintext index and security inverted index increases correspondingly. The construction of secure inverted index involves keyword encryption, logical address encryption, and encryption of inverted linked list nodes. Therefore, compared with the construction time of plaintext index, the construction time of secure inverted index will increase to a certain extent. However, the way to construct the security index is to use XOR arithmetic on the pseudorandom sequence generated by three PRFs for the original data. The operation is efficient, so the increment of the construction time of the secure inverted index is acceptable, which shows that the secure inverted index constructed in this article has high security while taking the efficiency of the traditional inverted index into account.

(b) Trapdoor generation time: In our scheme, trapdoor is composed of three pseudorandom sequences generated by PRF. The complexity of constructing trapdoor is $O(\lambda)$, so the time of constructing trapdoor is only related to the size of random seed $\lambda$, and independent of the total numbers of keywords $m$. As shown in Figure 3B, the time of generating trapdoor is almost unchanged with the increase of the number of keywords. Because scheme [11] needs to create four secret keys for each PRF, while our scheme just needs three secret key for the PRFs; in the trapdoor generation phase, our scheme is more efficient than scheme [11].

(c) Search time: According to the structural characteristics of inverted index, search time is mainly related to the size of keyword set and the number of documents containing the query keywords. The 10 000 keywords set constructed in this article, which appear in 1-32 documents. Because both our scheme and scheme [11] adopt the inverted index, both schemes can achieve effective search efficiency. As shown in Figure 3C, with the increase of the number of keywords, the search time increases slowly, and the scheme meets the user’s efficient search needs.

(d) Proof generation time: Suppose $x$ keywords are queried, and the theoretical amount of computation for generating $x$ keywords proof is $xE1$. As can be seen from Figure 3D, given fixed keyword set $n=10000$ and top-K = 10, the time of evidence generation is linearly related to the number of submitted keywords in the query phase. On the other hand, compared with literature,19 which requires the complete verification path of Merkle hash tree, the cloud server in this article generates proof by aggregating $x$ keywords and does not need to sign every query keyword separately, which greatly saves the transmission of verification data. As can be seen from Figure 3E, given fixed keyword set $n=10000$ and fixed submitted

| Scheme          | Verifiability | Result ranking | Search time | Verification efficiency |
|-----------------|---------------|---------------|-------------|-------------------------|
| Scheme [1]      | No            | No            | $O(|F|)$    | --                      |
| Scheme [2]      | No            | No            | $O(k')$     | --                      |
| Scheme [21]     | Yes           | Yes           | $O(m)$      | $2P + (2k + 2)E1$       |
| Our scheme      | Yes           | Yes           | $O(m)$      | $2P + kE1$              |

**TABLE 2** Functionality and efficiency comparison

Abbreviations: "-" no operation; "k" the number of selected ciphertext; "k" the number of documents associated with any given keyword; "m" the number of keywords.
keyword $x = 5$, the proof generation time of scheme$^{11}$ and our scheme have little change. Because the time of verification is related to the number of submitted keywords only.

(e) Verification time: The verification method of proposed scheme is divided into two processes. The first process is to verify the correctness of the search results, that is, to verify whether the results returned by the cloud server correspond to the query keywords specified by the user. The calculation amount is $2P + xe1$. In this process, the user uses the public key to compute pair operation of the proof returned by the cloud server and the keywords verification tags sent by the data owner on the group $G_1$. Comparing whether the results of these two operations are equal or not, if they are equal, then the cloud server correctly performs the retrieval operation. The second process is to verify the integrity of search results, that is, to verify whether the results returned by the cloud server are top-$K$ files, which only involves integer multiplication. As can be seen from Figure 3F, as the number of keywords queried by users increases, the verification time increases correspondingly. This is due to the pairing operation requires $x$ times exponential operations, but the increase is only at the level of milliseconds. For scheme [11] takes the RSA accumulator as the verification tool, and it involves the calculation of a verifiable matrix, it has a constant verification time.

7 CONCLUSION

Under the semitrusted and curious cloud storage server model, we suppose that the forged search results and the deception of user retrieval requests from the cloud storage servers may exist. We propose a ranked ciphertext search scheme supporting user to verify the retrieval results. First, the traditional inverted index structure is improved for building a secure inverted index to enhance the security of the scheme. Second, the bilinear mapping is used to generate validation tags for keywords to verify the correctness and reliability of the retrieval result. The experimental results show that the scheme is efficiency and can meet the security requirement of data user.

However, the proposed scheme does not support dynamic update operation of the index, which deserves further research in the next step.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant 61962005.
REFERENCES

1. Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. Paper presented at: Proceeding 2000 IEEE Symposium on Security and Privacy; 2000:44-55.

2. Kamara S, Papamanthou C. Parallel and dynamic searchable symmetric encryption. Paper presented at: Proceedings of the 2013 International Conference on Financial Cryptography and Data Security; 2013:258-274.

3. Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. J Comput Sec. 2011;19(5):895-934.

4. Wang C, Cao N, Ren K, Lou W. Enabling secure and efficient ranked keyword search over outsourced cloud data. IEEE Trans Parall Distrib Syst. 2011;23(8):1467-1479.

5. Sun W, Wang B, Cao N, et al. Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking. Paper presented at: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security; 2013:71-82. http://dx.doi.org/10.1145/2484313.2484322.

6. Cao N, Wang C, Li M, Ren K, Lou W. Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Trans Parall Distrib Syst. 2013;25(1):222-233.

7. Wong WK, Cheung DWI, Kao B, Mamoulis N. Secure KNN computation on encrypted databases. Paper presented at: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data; 2009:139-152. http://dx.doi.org/10.1145/1559845.1559862.

8. Jiang X, Yu J, Yan J, Hao R. Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data. Inf Sci. 2017;403:22-41.

9. Guo C, Chen X, Jie Y, Zhangjie F, Li M, Feng B. Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption. IEEE Trans Serv Comput. 2017;1:1–1. http://dx.doi.org/10.1109/tsc.2017.2768045.

10. Ge X, Yu J, Hu C, Zhang H, Hao R. Enabling efficient verifiable fuzzy keyword search over encrypted data in cloud computing. IEEE Access. 2018;6:45725-45739.

11. Liu Q, Tian Y, Wu J, Peng T, Wang G. Enabling verifiable and dynamic ranked search over outsourced data. IEEE Trans Serv Comput. 2019;1-1. http://dx.doi.org/10.101109/tsc.2019.2922177.

12. Chai Q, Gong G. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. Paper presented at: Proceedings of the 2012 IEEE International Conference on Communications (ICC); 2012:917-922. http://dx.doi.org/10.1109/ICC.2012.6364125.

13. Liu X, Yang G, Mu Y, Deng R. Multi-user verifiable searchable symmetric encryption for cloud storage. IEEE Trans Depend Sec Comput. 2019;1:1–1. http://dx.doi.org/10.1109/tsc.2018.2876831.

14. Li Y, Zhou F, Qin Y, Lin M, Xu Z. Integrity-verifiable conjunctive keyword searchable encryption in cloud storage. Int J Inf Sec. 2018;17(5):549-568.

15. Wang J, Chen X, Li J, Zhao J, Shen J. Towards achieving flexible and verifiable search for outsourced database in cloud computing. Future Generat Comput Syst. 2017;67:266-275.

16. Sun W, Liu X, Lou W, Hou YT, Li H. Catch you if you lie to me: efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data. Paper presented at: Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM); 2015:2110-2118. http://dx.doi.org/10.1109/INFOCOM.2015.7218596.

17. Zhu X, Liu Q, Wang G. A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing. Paper presented at: Proceedings of the 2016 IEEE Trustcom/BigDataSE/ISPA; 2016:845-851. http://dx.doi.org/10.1109/TrustCom.2016.0147.

18. Liu Q, Nie X, Liu X, Peng T, Wu J. Verifiable ranked search over dynamic encrypted data in cloud computing. Paper presented at: Proceedings of the 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS); 2017:1-6. http://dx.doi.org/10.1109/IWQoS.2017.7969156.

19. Azraoui M, Elkhiyaoui K, Önen M, Molva R. Publicly verifiable conjunctive keyword search in outsourced databases. Paper presented at: Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS); 2015:619-627. http://dx.doi.org/10.1109/CNS.2015.7346876.

20. Du R, Li M, Tian J, Wu W. Verifiable obfuscated keyword ciphertext retrieval scheme based on inverted index. J Softw. 2019;30(8):2362-2374.

21. Miao Y, Ma J, Wei F, Liu Z, Wang XA, Lu C. VCSE: verifiable conjunctive keywords search over encrypted data without secure-channel. Peer-to-Peer Netw Appl. 2017;10(4):995-1007.

22. Miao Y, Ma J, Liu X, et al. VCKSM: verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings. Pervas Mob Comput. 2017:40:205-219.

23. Liu Z, Li T, Li P, Jia C, Li J. Verifiable searchable encryption with aggregate keys for data sharing system. Future Generat Comput Syst. 2018:78:778-788.

24. Soleimanian A, Khazaei S. Publicly verifiable searchable symmetric encryption based on efficient cryptographic components. Des Codes Cryptogr. 2019;87(1):123-147.

25. Hui Z, Feng D, Zhang M, Hong C. A secure index against statistical analysis attacks. J Comput Res Dev. 2017:54(2):295-304.

26. De Caro A, Iovino V. JPBC: Java pairing based cryptography. Paper presented at: Proceedings of the 2011 IEEE Symposium on Computers and Communications (ISCC); 2011:850-855. http://dx.doi.org/10.1109/ISCC.2011.5983948.