Article | Received 31 December 2023; Accepted 1 June 2024; Published 14 June 2024
https://doi.org/10.55092/blockchain20240004

BC-RFMS: blockchain-based rankable fuzzy multi-keyword search scheme

Lixiang Zheng[1], Hanlin Zhang[1,*], Xinrui Ge[1], Jie Lin[2], Fanyu Kong[3] and Leyun Yu[4]

[1]College of Computer Science and Technology, Qingdao University, Qingdao, China
[2]School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, China
[3]School of Software, Shandong University, Shandong University, Jinan, China
[4]JIC IOT CO., LTD, Nanchang, China

* Correspondence author; E-mail: hanlin@qdu.edu.cn.

Abstract: The cloud provides convenient storage services for the vast amount of data generated by the Internet of Things (IoT), yet it also introduces security challenges such as data tampering and privacy breaches to IoT. While IoT encrypts sensitive data before storing it on cloud servers, it is unable to perform searches on encrypted data through these cloud servers. The advent of searchable encryption technology has successfully solved the problem of searching encrypted data, thereby protecting user data privacy to a certain extent. Most existing searchable encryption schemes support precise keyword searches, with the cloud servers providing encrypted search services. However, cloud servers may perform partial searches or fabricate some results due to various incentives, such as saving computational or storage resources. To address the aforementioned issues, we proposed BC-RFMS, a blockchain-based rankable fuzzy multi-keyword search scheme. Leveraging the immutable properties of blockchain, BC-RFMS ensured the accuracy of search results through smart contracts for fuzzy searching. Furthermore, the scheme utilized locality-sensitive hashing and Bloom filters to construct fuzzy keyword search indices, while employing term frequency-inverse document frequency (TF-IDF) for relevance score computation and ranking. We also proposed a block-by-block transfer algorithm to prevent surpassing the GasLimit when uploading data to the blockchain. To enhance storage security and reduce Gas consumption, encrypted data was stored on the InterPlanetary File System (IPFS) for distributed storage. Experimental analyses conducted on a blockchain test network validated the feasibility of the BC-RFMS scheme.

Keywords: blockchain; searchable encryption; rankable fuzzy keyword; Interplanetary File System (IPFS); IoT

1. Introduction

With the rapid advancement of the Internet of Things (IoT) and next-generation information technology, along with the widespread application of related services, an increasing number of individuals and enterprises are outsourcing data to the cloud to benefit from convenient and efficient storage and computing resources [1–3]. This has become a common phenomenon in the internet era. Although outsourcing data reduces costs and provides users with flexible cloud computing and storage services, cloud servers are inherently untrustworthy. Users do not have full control over their private data, resulting in inadequate protection of their privacy [4]. Therefore, maintaining the privacy of outsourced data faces a series of challenges, such as data tampering and theft of personal information. To some extent, encrypting personal data with existing encryption technologies (e.g., AES) before outsourcing it to cloud servers can

effectively protect the privacy of user data. However, users are unable to retrieve the desired data through the cloud server, which impacts the usability of user data. The most effective solution to this problem is the advent of searchable encryption technology.

Searchable encryption technology is an emerging method for ciphertext retrieval, allowing data to be encrypted and stored in the cloud without exposing plaintext, and then queries to be performed on the ciphertext. First introduced by Song *et al*. [5] in 2000 through a symmetric encryption-based scheme, it broke the deadlock of being unable to search through encrypted data, sparking widespread interest and research among scholars into the security of data outsourcing and privacy issues. Following this, a variety of searchable encryption schemes have been proposed to meet different requirements and functionalities, including those for single-keyword searches [6–10] and multi-keyword searches [11–16]. However, these schemes only support precise searches and do not accommodate fuzzy keyword searches. Consequently, the implementation of fuzzy keyword searching has garnered significant attention. To date, numerous search schemes [17–21] have been developed to address spelling errors or searches for approximate keywords.

The traditional fuzzy keyword searchable encryption schemes typically rely on centralized cloud servers to process data and queries, which can lead to single points of failure as well as security and privacy concerns. For instance, malicious acts by cloud servers, such as data tampering, could result in inaccurate search results, thereby harming user interests. To address these issues, many researchers have introduced blockchain into searchable encryption schemes, using the blockchain system to replace untrustworthy third-party cloud servers. Blockchain technology is characterized by its decentralization, public transparency, immutability and traceability. Blockchain technology offers an immutable data feature. Furthermore, blockchain-based searchable encryption schemes utilize smart contracts and cryptographic algorithms to more effectively protect users' query privacy and prevent keyword leakage to third parties. Smart contracts ensure that only authorized queries are executed, and that data during the execution process is not exposed externally. Therefore, blockchain-based fuzzy searchable encryption, with its unique security features, offers a more secure, private and reliable method for data searching. Compared to traditional schemes, it significantly improves the protection of user privacy, enhances data security and increases system availability.

In cloud-based searchable encryption schemes, most designs introduce blockchain primarily to ensure fairness, but do not implement search functionalities on the blockchain. If searching is to be conducted on the blockchain, the cost of data storage and data transmission becomes a critical issue. Due to the large volume of data, uploading it to the blockchain incurs significant Gas costs. Consider a smart agriculture project that utilizes multiple IoT sensors to collect data such as temperature and humidity, aiming to upload this data to the Ethereum blockchain to ensure its immutability and provide a reliable data source for agricultural research. However, if IoT devices generate hundreds of data records every minute, uploading directly to the blockchain would result in an extremely high transaction frequency. Each transaction incurs Gas fees, and any data upload could be rejected if it exceeds the GasLimit. Additionally, traditional approaches store vast amounts of data in the cloud, but system failures at the cloud server, such as hardware malfunctions, software bugs, or human errors, can disrupt data services. In extreme cases, significant network attacks like hacking could lead to data loss, causing irreparable damage to individuals and businesses. Therefore, traditional fuzzy keyword search schemes are generally not directly applicable in blockchain environments. How to implement fuzzy keyword search on blockchain and reduce cost consumption is an important research topic we are currently focusing on. To address the issues of Gas consumption and enable fuzzy search on the blockchain, we propose a secure and practical blockchain-based rankable multi-keyword fuzzy searchable encryption scheme, aimed at better compatibility improvements with the blockchain environment. The main contributions of this paper are as follows.

- To effectively reduce the Gas cost associated with data transfer to the blockchain, we have designed a block-by-block data transfer algorithm that avoids transaction cessation due to exceeding the GasLimit by transferring excessively large datasets at once.
- Our approach utilizes Locality Sensitive Hashing and Bloom filters to create fuzzy keyword indexes, effectively supporting multi-keyword fuzzy search on the blockchain. For keyword transformation, we have developed an improved trigram keyword transformation algorithm that efficiently prevents the occurrence of identical trigrams.
- To reduce storage costs and avoid single-point storage failures, encrypted files are stored on the InterPlanetary File System (IPFS) to achieve distributed storage, while only secure index data is stored on the blockchain.
- We conducted iterative experiments and evaluations with authentic datasets. Our comprehensive analysis of performance metrics and experimental outcomes illustrates the efficacy and practicality of our solution.

Organization: The remainder of this paper is structured as follows: Section 2 offers an exhaustive review and analysis of related work in searchable encryption. Section 3 outlines the background knowledge pertinent to our proposed solution. Section 4 details the system implementation model and threat model of our solution. In Section 5, we elaborate on the specific structure and algorithms of the solution and give a security analysis of the solution at the end. Section 6 is a discussion of the adaptation of this scheme to IoT scenarios. Section 7 discusses the experimental methodology and performance evaluation. Lastly, Section 8 delivers a succinct and conclusive summary of the paper and describes future work.

2. Related work

Song *et al*. [5] were the first to study cryptographic techniques supporting search over encrypted data, pioneering a new direction in cryptography known as “Searchable Encryption." Since this seminal paper was published, this area has garnered sustained close attention from both the academic and industrial communities, establishing it as a critical field of inquiry. An increasing number of searchable encryption schemes based on diverse search functionalities continue to emerge. These include boolean and conjunctive searches [11–16], ranked searches [22–28], and fuzzy searches [17–21], among others, demonstrating the field’s evolution and diversification.

In recent years, many researchers have dedicated efforts to design mechanisms that support effective matching for exact keywords. However, most schemes tend to overlook mismatches in search results caused by misspelled keywords during the search process, consequently leading to users receiving incorrect file data. To tackle this, Li *et al*. [17] introduced a fuzzy keyword search in cloud environments, utilizing edit distance for keyword similarity and wildcard technology for generating keyword sets, thus providing users with closely matched documents. However, this method, limited to single keyword searches, incurs considerable storage overhead. Wang *et al*. [29] developed a private Trie-based search index, using edit distance for constant-time similarity searches. Kuzu *et al*. [30] utilized Jaccard distance and minhash for error-tolerant multi-keyword searches. Another approach by Wang *et al*. [18] combined LSH functions and Bloom filters, employing Euclidean distance to efficiently manage multi-keyword searches. They further introduced a scheme [19] supporting range searches and document ranking using a double-layer Bloom filter and a score table. Fu *et al*. [20] advanced a multi-keyword ranked fuzzy search building upon the approach in [29], enhancing keyword transformation and correlating keyword weight with document relevance. Ge *et al*. [31] designed a linked-list based index for exact and fuzzy searches, reducing storage needs. Zhang *et al*. [32] used edit distance and a binary tree index for enhanced search efficiency, though limited to single keyword queries. Zhong *et al*. [21] proposed a balanced binary tree index for multi-keyword fuzzy searches, capable of returning top results efficiently. The aforementioned schemes have implemented and improved to varying extents in terms of

search structure and matching algorithms for fuzzy keyword search. However, they all rely on cloud servers to execute fuzzy keyword searches and other functions.

Blockchain, characterized by decentralization, transparency, and immutability, substitutes the cloud server in traditional searchable encryption schemes, thus broadening the research horizon for scholars. Li *et al*. [33] addressed threats from malicious cloud servers by storing ciphertext data and security indexes on the blockchain, introducing two searchable encryption schemes based on data size. This approach, however, increases blockchain storage costs to varying extents due to the need to store all encrypted data on-chain. Hu *et al*. [34] innovated ciphertext search by replacing cloud servers with blockchain smart contracts, ensuring search result correctness and fairness. Expanding on this, Chen *et al*. [35] combined Boolean search with blockchain to develop a multi-keyword searchable encryption scheme. Following a similar path [33], Li *et al*. [36] proposed an optimized blockchain-based searchable encryption model to improve usability. Zhang *et al*. [37] offered a scheme featuring bidirectional authentication and a digitally signed search index, enhancing result integrity but raising user computational burdens due to signature verification. While these schemes enable precise keyword searchable encryption via blockchain, they neglect fuzzy keyword search capabilities. Yan *et al*. [38] introduced a blockchain-based fuzzy keyword search method using edit distance to generate fuzzy keyword sets, conducting searches via cloud servers with integrated blockchain and RSA accumulators for result verification. Chakraborty *et al*. [39] created a scheme employing locality-sensitive hashing and Bloom filters for blockchain-based fuzzy keyword search, yet it lacks functionality for result sorting.

We performed a comparative analysis to assess the functionality of our BC-RFMS scheme against existing schemes, as detailed in Table 1. This analysis reveals that while all reviewed schemes support single keyword search, only the approach outlined in schemes [18, 35, 39–41] facilitate simultaneous multi-keyword search. Additionally, schemes [18, 39, 41] uniquely offer support for fuzzy keyword search. Noteworthy is the method employed by reference schemes [18, 38, 40, 41], where search operations are executed by the cloud server, and schemes [18, 40, 41], which introduces result sorting capabilities. In terms of data storage, schemes [18, 38–41] utilize cloud server storage, contrastingly, schemes [34, 35] explicitly mention cloud server-based data storage, and scheme [34] provides a somewhat vague explanation, albeit mentioning IPFS. Compared to these schemes, our BC-RFMS scheme showcases the functionalities specified in the table.

Table 1. Comparison with other schemes.

Function / Scheme	Single-keyword	Muti-keyword	Fuzzy search	Result ranking	Search on Blockchain	Location of data storage
Yan *et al*. [38]	✓	×	×	×	×	Cloud Server
Hu *et al*. [34]	✓	×	×	×	✓	Cloud Server/IPFS
Chen *et al*. [35]	✓	✓	×	×	✓	Cloud Server/IPFS
Xia *et al*. [40]	✓	✓	×	✓	×	Cloud Server
Fu *et al*. [41]	✓	✓	✓	✓	×	Cloud Server
Wang *et al*. [18]	✓	✓	✓	✓	×	Cloud Server
Chakraborty P. S. *et al*. [39]	✓	✓	✓	×	✓	Cloud Server
Our BC-RFMS	✓	✓	✓	✓	✓	IPFS

3. Preliminaries

3.1. Blockchain and smart contracts

Blockchain is a decentralized, distributed ledger technology that links transactions in blocks in chronological order, forming an immutable chain. Each block contains the hash value of the preceding block, ensuring data integrity and security. The fundamental concepts of this technology include consensus algorithms, distributed networks, and decentralized characteristics, which together ensure the reliability and transparency of the blockchain.

Blockchain's core concept is the distributed ledger, a database shared, replicated and synchronized among the members of a network. This distributed ledger records transactions between network participants, reducing the time and expenses incurred from reconciling different ledgers.

The advent of smart contracts signifies the arrival of the Blockchain 2.0 era, epitomized by Ethereum. Ethereum (ETH), created by Vitalik Buterin, is a cryptocurrency that facilitates the operation of smart contracts and decentralized applications (Dapps) on its network. Smart contracts, underpinned by blockchain technology, are autonomous programs that enforce contractual terms, delineating the stipulations and prerequisites of the agreement. Smart contracts are coded to execute on the blockchain automatically when specific conditions are met, eliminating the need for intermediaries or trusted third parties. These contracts often leverage Turing-complete programming languages, such as Solidity, to facilitate flexible and complex contractual logic. This innovation not only streamlines transaction processes but also significantly enhances the transparency and efficiency of contractual engagements.

Blockchain and smart contracts, as innovative technologies for distributed ledgers and automated contract execution, have attracted considerable interest across multiple sectors, including finance, supply chain, healthcare, and more. These technologies promise to create decentralized, secure and transparent transaction environments, revolutionizing the way transactions and agreements are conducted.

3.2. Bloom filter

Bloom filters (BF) are efficient data structures primarily utilized for rapid searching within large-scale datasets. A Bloom filter consists of an expansive binary vector coupled with a collection of random mapping functions, aimed at verifying the presence of an element in a set. The construction of a Bloom filter begins with the initialization of a bit array of length m, setting all bits to 0. Subsequently, we define k hash functions. Upon inserting an element, it is mapped through these k hash functions, setting the corresponding bit array positions to 1. When querying for an element, the query element undergoes the same k hash function mappings to check the corresponding bit array positions. If all the mapped positions are 1, the query element is considered potentially present; if any position is 0, the element is definitively absent. Compared to other data structures, Bloom filters offer advantages such as low time complexity and reduced storage space requirements. However, they also present challenges including a certain rate of false positives and difficulties in deletion. Figure 1 illustrates an example of a Bloom filter.

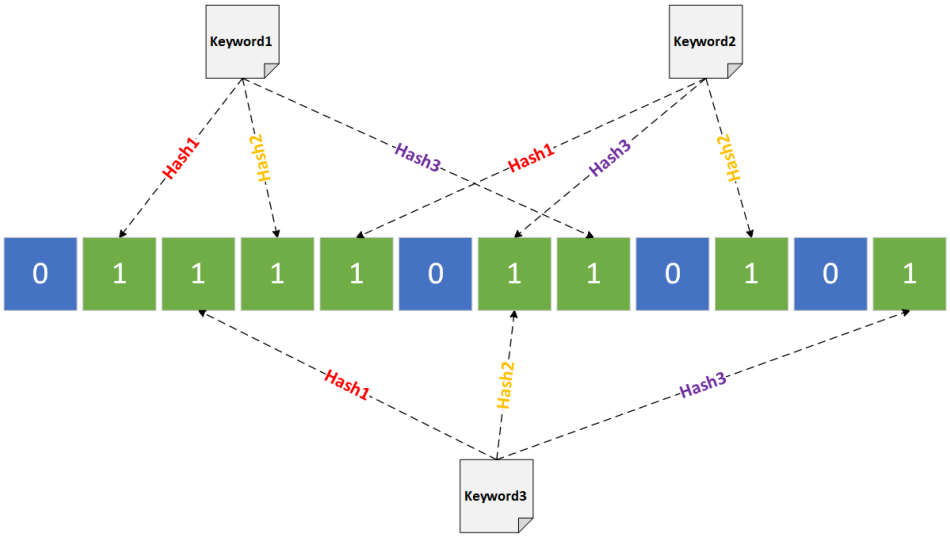


Figure 1. Bloom filter.

3.3. Locality-sensitive hashing

Locality-sensitive hashing (LSH) is recognized as an effective algorithm designed for efficient nearest neighbor searches within extensive high-dimensional datasets. It employs a specialized hash function that probabilistically assigns two highly similar data points to the same “bucket", yielding identical hash values, whereas dissimilar data points have a significantly lower probability of sharing the same “bucket". A hash function family, represented as H, is deemed (d_1, d_2, p_1, p_2)-sensitive f it meets the following criteria for any $h \in H$:

(1) For any two data points O_1 and O_2, if the distance $d(O_1, O_2) < d_1$, then the probability $Pr[h(O_1) = h(O_2)] \geq p_1$;

(2) If $d(O_1, O_2) > d_2$, then $Pr[h(O_1) = h(O_2)] \leq p_2$.

Here, O_1 and O_2 symbolize any two data points, with $d(O_1, O_2)$ indicating the distance between them. The constants d_1 and d_2 are predetermined, with $d_1 < d_2$, and p_1 and p_2 are probability constants ranging from 0 to 1.

3.4. Trigram language model

The trigram language model is a variant of the N-Gram model. It operates by applying a sliding window of size three bytes across the text content, resulting in a sequence of byte trigrams. For example, the trigram sequence for “blockchain" would be $\{blo, loc, ock, ckc, kch, cha, hai, ain, inb, nbl\}$. This method is utilized to process keyword sets, catering to the needs of fuzzy keyword search queries. It’s important to note that the keywords must consist of at least three characters. Concurrently, we refine the segmentation algorithm to enhance search precision by accounting for the recurrence of identical subkeywords, a detail elaborated in the scheme’s specific construction section.

4. System overview

In this section, we delineate the system model and threat model of our proposed scheme.

4.1. System model

The system model comprises four principal entities: Data owner (DO), Data user (DU), Blockchain (BC) and the InterPlanetary File System (IPFS), as illustrated in Figure 2. The core functions of each entity are outlined as follows:

(1) **Data owner (DO)**: The Data owner is responsible for creating and managing their data assets, such as collections of plaintext documents. They have the authority to control user access and queries.

(2) **Data user (DU)**: Data users are entities seeking access to and querying data housed within the system. They submit requests for data access to the DO and retrieve the authorized data upon approval.

(3) **Blockchain (BC)**: The Blockchain serves to store the index data of the DO and facilitates querying services for the DU. Data searches are executed through smart contracts. As a decentralized and immutable ledger, the Blockchain meticulously logs transactions and interactions between the DO and DU, guaranteeing the operations’ reliability and security.

(4) **InterPlanetary File System (IPFS)**: The InterPlanetary File System is a peer-to-peer distributed file system designed for decentralized storage and retrieval of files. It ensures efficient and resilient data storage, enabling the system to distribute data across various nodes. Encrypted data files secured by the DO are stored on IPFS, facilitating user access to the data.

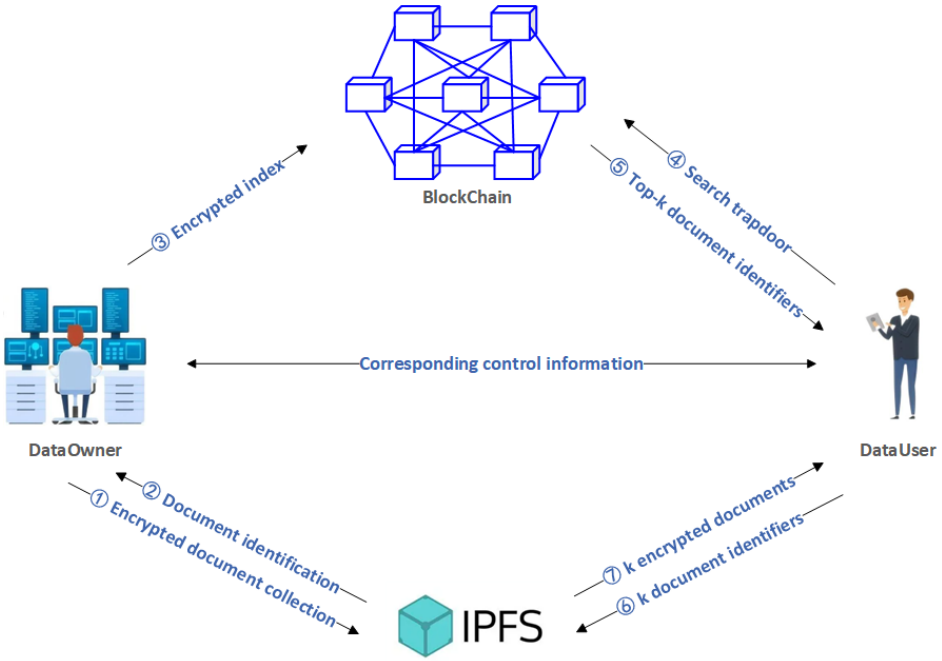


Figure 2. System model.

Specifically, the workflow of our system model is as follows:

Step 1. The data owner (DO) initially encrypts the plaintext document collection and dispatches it to the InterPlanetary File System (IPFS).

Step 2. Upon receipt, IPFS archives the encrypted collection and assigns unique document identifiers, which are subsequently relayed back to the DO.

Step 3. The DO then undertakes data preprocessing, leveraging the document identifiers and the encrypted documents to produce encrypted index data files, which are then uploaded to the Blockchain (BC).

Step 4. When a data user (DU) seeks data access, the DO, following authorization, transmits the necessary control information to the DU via a secure channel. Armed with this control information and their query keywords, the DU crafts search tokens and forwards them to the BC.

Step 5. The BC, utilizing smart contracts, processes the search request. Upon completion of the search, it furnishes the DU with the encrypted identifiers of the top-k encrypted documents pertinent to the fuzzy search terms.

Step 6. The DU, using the decryption key, decrypts these identifiers and transmit the k decrypted plaintext document identifiers to IPFS to retrieve the specified encrypted documents.

Step 7. IPFS locates and delivers the files based on the decrypted identifiers back to the DU. Finally, the DU decrypts the received files with the decryption keys, accessing the top-k plaintext documents that closely match their fuzzy search queries.

4.2. Threat model

Due to the characteristics of blockchain consensus mechanisms, the blockchain can execute data search operations accurately. We assume the presence of potentially malicious adversaries who can analyze transaction activities within the blockchain to obtain private information. Consequently, our solution addresses two types of threat models: the Known Ciphertext Model and the Known Background Model.

In the Known Ciphertext Model, the malicious adversary can only observe transactional activity on the blockchain, which includes encrypted indices, query trapdoors and search results. In the Known Background Model, the malicious adversary possesses statistical

information and query frequency capabilities, allowing for the analysis of search results returned from the blockchain's transactional activities.

5. The proposed BC-RFMS scheme

In this section, we describe the scheme descriptions and the design details. In our scheme, the search operations on data are executed by smart contracts and data documents are stored on IPFS, ensuring the reliability of search results.

5.1. Key generation

The data owner (DO) invokes a symmetric encryption algorithm using the security parameter λ to generate the encryption key K. Furthermore, two $(m+m') \times (m+m')$ reversible matrices, M_1 and M_2, and a $(m+m')$-bit vector $\overrightarrow{S}$ are randomly generated. $\overrightarrow{S}$ belongs to the set $\{0,1\}^{m+m'}$ and is determined by the index length m. m' denotes the number of phantom terms. Consequently, the DO holds a key set, $SK = \left\{K, M_1, M_2, \overrightarrow{S}\right\}$, encompassing all necessary cryptographic elements.

5.2. Document preprocessing

(1) **Document data encryption.** The data owner (DO) encrypts each document d_i in the plaintext document collection $D = \{d_1, d_2, \cdots, d_n\}$ utilizing the encryption key K, producing an encrypted document collection $C = \{c_1, c_2, \cdots, c_n\}$. Following encryption, the DO transmits C to the InterPlanetary File System (IPFS), which then archives the collection and issues a corresponding set of document identifiers $L = \{l_1, l_2, \cdots, l_n\}$. Subsequently, the DO encrypts L, creating EL, the encrypted collection of document identifiers. This process corresponds to Steps 1 and 2 in Figure 2.

(2) **Keyword extraction.** For a specified plaintext document collection D, the data owner (DO) derives keywords w_i from each document within D. Utilizing the Porter stemming algorithm, the DO extracts stem keywords kw_{st_i} for every keyword, resulting in the stemmed keyword set KW_{st}. The DO then employs the TF-IDF method to compute the Term Frequency (TF) value for each keyword, facilitating an enhanced analysis of the keyword-document relevance. TF denotes the frequency of occurrence of a keyword in a document, i.e., the number of a keyword in a given document divided by the total number of all keywords in that document. Note that the TF-IDF rule is a relatively mature weighted statistical technique widely used in the fields of information retrieval and text mining [42].

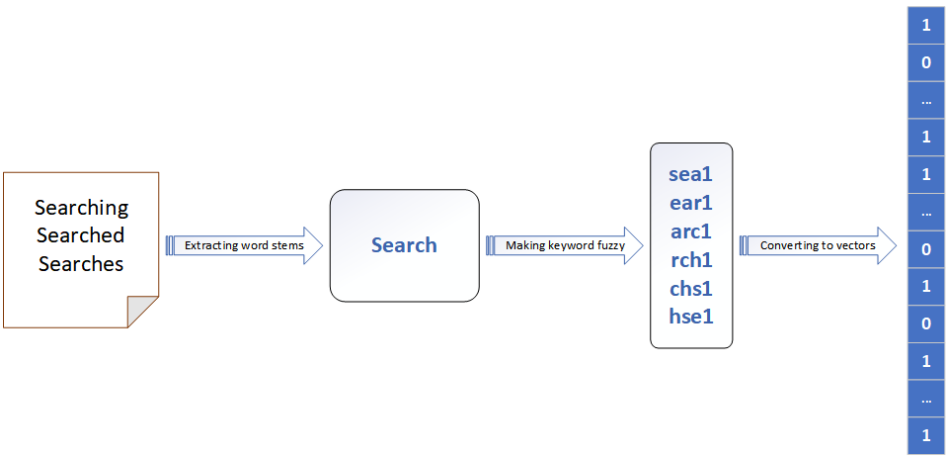


Figure 3. Document preprocessing.

(3) **Keyword transformation.** To facilitate fuzzy searches for users, the DO implements fuzzy processing on the keywords, employing an enhanced trigram keyword transformation technique. This method transforms each keyword into a trigram set TS, mindful of potential trigram repetitions, as detailed in Algorithm 1. For example, the trigram set TS for "banana" would comprise $\{ban1, ana1, nan1, ana2, nab1, aba1\}$. The DO then translates TS into a fixed-length binary vector $\overrightarrow{V}_{bin}$, with this data preprocessing procedure illustrated in Figure 3.

Algorithm 1: Generate Binary Vectors based on Triples

Input: The stemming keyword set KW_{st};

Output: The binary vector $\overrightarrow{V}_{bin}$ based on triples;

for *each stemming keyword* kw_{st_i} *in* KW_{st} **do**
- Generate a vector $\overrightarrow{V}_{st_i}$ and initialize all of its positions with the value 0 in 3×2^4 bits $\{0,1\}$ vector ;
- Compute the length of the keyword kw_{st_i} $l = Length(kw_{st_i})$;
- Generate the count vector $\overrightarrow{V}_{co}$ of length l and set the value of each of its bits to 1;
- **for** j = *0 to* l **do**
  - **for** k = *1 to* $j-1$ **do**
    - $Temp_1 = kw_{st_i}[k \mod l] + kw_{st_i}[(k+1) \mod l] + kw_{st_i}[(k+2) \mod l]$;
    - $Temp_2 = kw_{st_i}[j \mod l] + kw_{st_i}[(j+1) \mod l] + kw_{st_i}[(j+2) \mod l]$;
    - **if** $Temp_1 == Temp_2$ **then**
      - $\overrightarrow{V}_{co}[j]$ += 1;
  - $TS[j] = kw_{st_i}[j \mod l] + kw_{st_i}[(j+1) \mod l] + kw_{st_i}[(j+2) \mod l] + \overrightarrow{V}_{co}[j]$;
- **for** *each* $TS[j] \in TS$ **do**
  - Set the value of the corresponding position of $\overrightarrow{V}_{st_i}$ to 1;

Return a ternary-based binary vector $\overrightarrow{V}_{bin} = \overrightarrow{V}_{st_i}$.

5.3. *Index creation*

For each document d_i, the data owner first creates an m-bit Bloom filter BF_i, initializing all bits to 0. For each keyword w_j in d_i, the binary vector $\overrightarrow{V}_{bin_j}$ serves as input to N locality-sensitive hashing (LSH) functions, which assign the vector to specific positions in BF_i and set their values to $TF_{i,j}/N$. $TF_{i,j}$ represents the term frequency of w_j in d_i. In cases where different keywords converge on the same Bloom filter position, the system averages the values at these overlapping positions. Finally, BF_i is augmented from m to $m+m'$ bits, with the addition of m' randomly generated numbers to enhance security.

To safeguard the privacy of the index, the DO encrypts each Bloom filter BF_i into two separate entities, BF_i' and BF_i''. The encryption is executed as follows: according to a random vector $\overrightarrow{S}$, for each element where $\overrightarrow{S}[j] = 0$, both $BF_i'[j]$ and $BF_i''[j]$ are set equal to $BF_i[j]$. Conversely, if $\overrightarrow{S}[j] = 1$, $BF_i'[j]$ is assigned a random number δ, and $BF_i''[j]$ is set to $BF_i[j]$ - δ. Subsequently, the DO utilizes the transposed matrices of reversible matrices M_1 and M_2 to encrypt BF_i' and BF_i'', respectively, producing the encrypted index $EI_i = \{\mathbf{M}_1^T BF_i', \mathbf{M}_2^T BF_i''\}$. This process is detailed in Algorithm 2.

Upon generating the encrypted index set EI, the DO is tasked with uploading it to the blockchain. To minimize transmission gas costs and prevent breaching the GasLimit which could result in transaction failure, a block-by-block transmission strategy is implemented, as detailed in Algorithm 3. This approach uses a Pseudorandom Function (PRF) to obscure

the transmitted data, defined by $F : \{0,1\}^{\lambda} \times \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda}$. Firstly, the DO blinds and encapsulates the elements of EI and EL into a list B using function F, then divides list B into t blocks, each labeled as B_i. The DO then sends k_1 and initiates t transactions to the blockchain, each transmitting a block B_i. Then smart contracts on the blockchain are responsible for receiving these block data, parsing them into a empty dictionary Ω and storing them as key-value pairs. The smart contract uses k_1 to parse all block data, ultimately reconstructing the complete encrypted index set EI and the encrypted document identifier set EL. This process is illustrated in Figure 4. The operations in this subsection correspond to Step 3 in Figure 2.

Algorithm 2: Build Index

Input: The set of documents D and the set of keys SK;
Output: The encrypted index set EI;

for *each document* d_i *in* D **do**
  Generate an m-bit bloom filter BF_i and initialize each bit to 0;
  for $w_j \in d_i$ **do**
    Generate the corresponding binary vector $\overrightarrow{V}_{bin_j}$;
    The vector V_{bin_j} is hashed by N LSH functions to determine its corresponding position in the bloom filter BF_i, and the value at that position is set to $TF_{i,j}/N$, namely $BF_i[LSH\_FUNCTION_k(\overrightarrow{V}_{bin_j})] = TF_{i,j}/N, k \in [1, n]$.
  Fill the number of BF_i bits to $(m+m')$ bits;
  Initialize two $(m+m')$ bit vectors BF_i' and BF_i'';
  for *1 to* $(m+m')$ **do**
    if $\overrightarrow{S}[j] = 0$ **then**
      $BF_i'[j] = BF_i''[j] = BF_i[j]$;
    else
      $BF_i'[j] = \delta$;
      $BF_i''[j] = BF_i[j]$ - δ;
  Generate the encrypted index $EI_i = \{\mathbf{M}_1^T BF_i', \mathbf{M}_2^T BF_i''\}$ and add it to the encrypted index set EI;
The encrypted index set EI is divided into t transactions and sent to the blockchain.

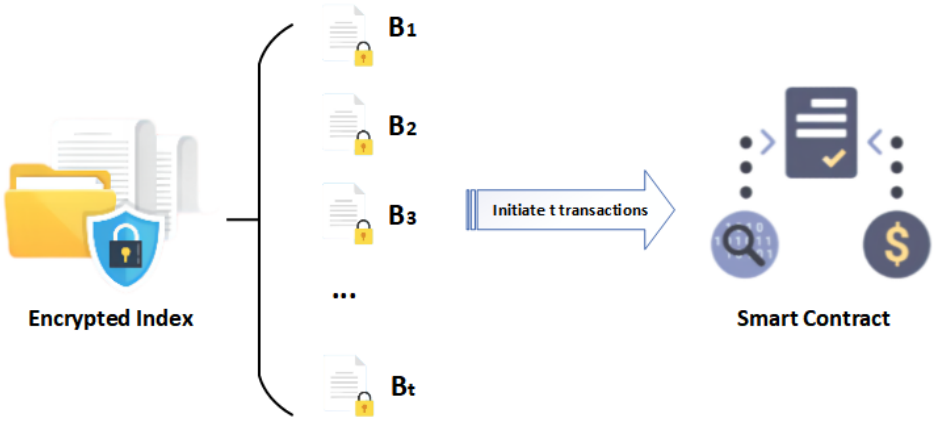


Figure 4. Block-block transmission.

Algorithm 3: Block-by-block Transmission

The data owner initializes an empty list B and sets c to 0;
for *each encrypted index* EI_i *in* EI **do**
  $k_1 \leftarrow F(K, EL_i)$;
  $\tilde{e} \leftarrow EI_i || EL_i$;
  $l \leftarrow F(k_1, c)$;
  c ++;
  add <l, $\tilde{e}$> to the list B;
The data owner sends k_1 and divides the list B into t blocks, i.e. B_i, $1 \leqq i \leqq t$, and then sends them to the blockchain in order by t transactions;
Smart contract parses the received data: initializes an empty dictionary Ω and store $\{key : l, value : \tilde{e}\}$ into Ω; sets c to 0;
for *each element in* Ω **do**
  $l \leftarrow F(k_1, c)$;
  $\tilde{e} \leftarrow Get(\Omega, l)$;
  c ++;
  parse each $\tilde{e}$ and get the corresponding encryption index EI_i and encrypted document identity EL_i;
Return the encrypted index set EI and the encrypted document identity set EL.

5.4. Trapdoor generation

Data users submit search requests to the data owner, who authorize and send relevant control information (e.g., the key set SK) to the DU. Subsequently, the DU preprocess the keywords to be queried based on given requirements. The preprocessing process includes keyword extraction and keyword transformation, similar to steps (2) and (3) of the document collection preprocessing. In the transformation phase, each keyword is converted into a binary vector $\overrightarrow{QV}_{bin}$, utilizing its trigram set.

The DU constructs an m-bit Bloom filter BF_q, initializing all bits to 0. The binary vector $\overrightarrow{QV}_{bin}$, representing the query keywords, serves as input for N LSH functions. These functions map the vector to specific positions within BF_q, where values are assigned based on the Inverse Document Frequency (IDF) of each keyword. IDF is the inverse of the ratio of the number of documents containing a certain keyword to the total number of all documents in the collection. Subsequently, the DU expands BF_q to $m + m'$ bits, randomly setting m'' bits to 1 within the additional m' space, with the rest maintained at 0.

To safeguard trapdoor privacy, the data user encrypts the query keyword Bloom filter BF_q into two separate entities, BF'_q and BF''_q. The encryption methodology is as follows: if $\overrightarrow{S}[j] = 1$, then both $BF'_q[j]$ and $BF''_q[j]$ are set to $BF_i[j]$. Conversely, if $\overrightarrow{S}[j] = 0$, $BF'_q[j]$ is assigned a random number δ, and $BF''_q[j]$ is set to $BF_q[j]$ - δ. Subsequently, BF'_q and BF''_q are encrypted via the inverse matrices of reversible matrices M_1 and M_2, producing encrypted search keyword vectors $\overrightarrow{EQV} = \{\mathbf{M}_1^{-1} BF'_q, \mathbf{M}_2^{-1} BF''_q\}$. The search trapdoor $TD = \left\{k, \overrightarrow{EQV}\right\}$ is then transmitted to the blockchain, as delineated in Algorithm 4. The operations in this subsection correspond to Step 4 in Figure 2.

Algorithm 4: Trapdoor generation

Input: The set of keywords QW for the query;
Output: The search trapdoor TD;

Generate an m-bit bloom filter BF_q and initialize each bit to 0;
for qw_j *in* QW **do**
  Generate the corresponding binary vector $\overrightarrow{\boldsymbol{QV}}_{bin}$;
  The vector $\overrightarrow{\boldsymbol{QV}}_{bin}$ is hashed by N LSH functions to determine its corresponding position in the bloom filter BF_q, and the value at that position is set to IDF_{qw_j}/N, namely $BF_q[LSH\_FUNCTION_k(\overrightarrow{\boldsymbol{QV}}_{bin})] = IDF_{qw_j}/N, k \in [1, n]$.
  Fill the number of BF_q bits to $(m+m')$ bits;
  Initialize two $(m+m')$ bit vectors BF_q' and BF_q'';
  for *1 to* $(m+m')$ **do**
    if $S[j] = 1$ **then**
      $BF_q'[j] = BF_q''[j] = BF_i[j]$;
    else
      $BF_q'[j] = \delta$;
      $BF_q''[j] = BF_q[j]$ - δ;
  Generate encrypted search keyword vectors $\overrightarrow{\boldsymbol{EQV}} = \{\mathbf{M}_1^{-1}BF_q', \mathbf{M}_2^{-1}BF_q''\}$;
Finally, the search trapdoor $TD = \left\{k, \overrightarrow{\boldsymbol{EQV}}\right\}$ is sent to the blockchain.

5.5. *Data search*

The smart contract on the blockchain executes this procedure. This procedure corresponds to Step 5 in Figure 2. It performs an inner product operation between each encrypted index EI_i and the encrypted search keyword vectors $\overrightarrow{\boldsymbol{EQV}}$ to compute a relevance score $RSco_i$, where $RScoi = EI_i \cdot \overrightarrow{\boldsymbol{EQV}} = \left(\mathbf{M}_1^T BF_i'\right) \cdot \left(\mathbf{M}_1^{-1} BF_q'\right) + \left(\mathbf{M}_2^T BF_i''\right) \cdot \left(\mathbf{M}_2^{-1} BF_q''\right) = BF_i' BF_q' + BF_i'' BF_q'' = BF_i \cdot BF_q$. Based on this operation, the contract identifies and returns the top-k documents with the highest relevance scores to the data user, as detailed in Algorithm 5. For a more intuitive understanding of the index creation, trapdoor generation and search phases, refer to Figure 5.

Algorithm 5: Blockchain search

Input: The encrypted index set EI and trapdoor TD;
Output: The set $RSet$ of search results containing k ciphertext identities;

The smart contract initializes an empty set $RSet$ and a threshold th;
for EI_i *in* EI **do**
  Compute each relevance score $RSco_i = EI_i \cdot \overrightarrow{\boldsymbol{EQV}}$;
  if $RSco_i > th$ **then**
    Put the element $\{L_i, RSco_i\}$ in the set $RSet$;
    if $|RSet| > k$ **then**
      Sort $RSet$;
      Delete $(|RSet| - k)$ entries from $RSet$;
      $th = \min\{RSet\}$;
Return $RSet$.

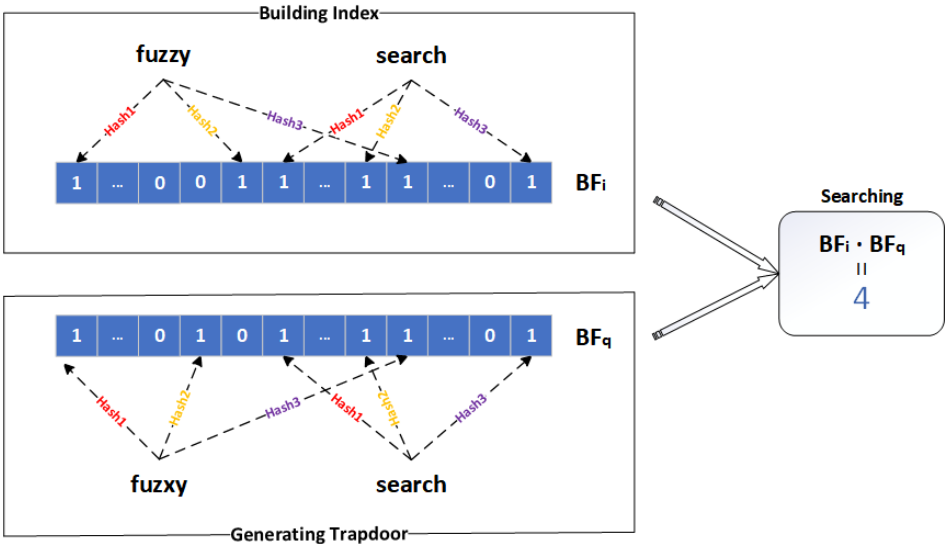


Figure 5. Process of operation.

5.6. Document decryption

Upon receiving the k encrypted document identifiers, the data user decrypts them and forwards these identifiers to the InterPlanetary File System (IPFS). IPFS retrieves the corresponding k encrypted documents using these identifiers and sends them back to the data user. Subsequently, the user requests decryption keys from the DO. After authenticating the user successfully, the DO transmits the decryption keys to the data user. Finally, the data user utilizes the decryption key to decrypt the k encrypted documents, thereby obtaining the top-k plaintext documents most pertinent to the query keywords. This process corresponds to Steps 6 and 7 in Figure 2.

Security analysis. We conduct a security analysis of the proposed solution based on the outlined threat models.

- **Document confidentiality.** In this approach, the data owner encrypts documents using a symmetric encryption algorithm with the symmetric key K and sends the encrypted document collection to IPFS. Data users request access authorization through a secure channel from the data owner, who must send the key K to the data users before they can decrypt the files. In other words, without the key K, a malicious adversary cannot decrypt the document data, thus ensuring the confidentiality of the document data.
- **Confidentiality of indexes and query trapdoors.** Under the Known Ciphertext Model, a malicious adversary may observe encrypted index information, encrypted document identifiers and encrypted query trapdoors through block information. However, without the keys, it is challenging to analyze any information. In this solution, EI_i and $\overrightarrow{\boldsymbol{EQV}}$ are obfuscated vectors, making it impossible for a malicious adversary to deduce the original vectors BF_i and BF_q without the key set SK. Reversible matrices M_1 and M_2 are random Gaussian matrices, and the use of dummy items in the matrices makes the transformation matrices more challenging to compute, thereby increasing the difficulty for a malicious adversary. Under the Known Background Model, since data users upload encrypted query trapdoors and the blockchain returns encrypted document identifiers, it is hard for a malicious adversary to perform statistical analysis on transaction information on the blockchain without the keys. Hence, this solution effectively protects the confidentiality of indexes and query trapdoors.
- **Unlinkability of queries.** Under the Known Ciphertext Model, this solution introduces m' random numbers so that the same search request will generate different query vectors. They make it difficult for a malicious adversary to analyze the query trapdoors, thereby ensuring the unlinkability of queries. In the Known Background Model, since the

introduction of random values results in different distributions of relevance scores, a malicious adversary cannot deduce trapdoor information through statistical capabilities from the blockchain's transaction information.

6. Discussion

This study introduces BC-RFMS, a blockchain-based searchable encryption scheme, optimized for use in the Internet of Things (IoT). The fusion of IoT with blockchain technology and searchable encryption catalyzes a myriad of novel research avenues and practical applications, including smart home systems, smart cities, supply chain management and healthcare. Taking the healthcare scenario as an example, medical devices generate a vast amount of sensitive personal health information such as medical history and diagnostic reports. Therefore, we are able to use this personal health data as the dataset for our scheme. By leveraging the InterPlanetary File System (IPFS) for distributed storage of encrypted personal information, our approach indexes this data and stores the encrypted index information on the blockchain to ensure data security. The blockchain-based searchable encryption model utilizes the immutable nature of the blockchain to guarantee that data records and search results cannot be altered, ensuring that only authorized users can search and access the data. Authorized medical institutions can access patient data and retrieve the most relevant past health and medical records through fuzzy keyword search. They then analyze encrypted health records to enhance diagnostic efficiency and research quality. This methodology not only enhances diagnostic and research capabilities but also promotes the secure sharing of medical information, thereby optimizing the utility and quality of personal health data services. However, this scheme may be inefficient in handling large-scale data in IoT environment, potentially causing system delays and other issues. This highlights a scalability problem that will be a primary focus of our future research. Therefore, in the subsequent experimental testing, we used a small-scale dataset to evaluate the proposed scheme.

7. Experiments and performance evaluation

In this section, we present the feasibility and performance of our BC-RFMS scheme through experimental validation and analysis. The experimental setup was conducted on a system running Ubuntu 20.04 LTS, powered by an Intel(R) Core(TM) i7-11700T CPU at 1.4 GHz with 16 GB of RAM. We implemented our smart contracts in Solidity and deployed them on the Ethereum test network, Spolia. For our experiments, we utilized the real-world Enron email dataset, which was pre-processed using Python. We configured the scheme with three locality-sensitive hashing (LSH) functions ($n = 3$) and set the length of indexes and trapdoors (m) to 500.

Our scheme underwent an experimental comparison with schemes [18] and [39], focusing on stages that include index creation, trapdoor generation and blockchain-based search.

Index creation phase: In this phase, the main time expenditure stems from generating the plaintext index and its subsequent encryption. Encrypting the index with the matrix requires $O(m^2n)$ multiplications, where m is the index length and n represents the document count. As depicted in Figure 6, the index creation process for the BC-RFMS scheme shows a gradual rise in time consumption with an increasing number of keywords. This is due to differences in the specific architecture of fuzzy keyword search. Although both the comparative scheme and our scheme use Bloom filters to construct indexes and query trapdoors, in our scheme we have improved the method for fuzzy keyword processing. Additionally, we have standardized the parameters according to the comparative scheme. Similarly, the subsequent trapdoor generation process follows the same principle.

Trapdoor generation phase: During this phase, the time complexity is $O(m^2)$, with m denoting the trapdoor length. The generation of the trapdoor involves multiplying the search

token by matrix encryption. Figure 7 illustrates that the time required for trapdoor generation is dependent on the number of query keywords, showing a gradual increase as the keyword count rises. This pattern is consistent with observations from the index generation phase, with time consumption maintaining stability around the 25 ms mark.

Search phase on the blockchain: In this phase, the time cost is attributed to the execution of the inner product operation between the index and trapdoor on the smart contract, exhibiting a time complexity of $O(mn)$. Both scheme [39] and our scheme perform searches on the blockchain; however, the scheme [39] has not tested the search efficiency on the blockchain. Additionally, the blockchain test networks (Ropsten and Rinkeby) used in scheme [39] are now deprecated. We conducted our tests on the currently available blockchain test network, Sepolia. Consequently, in terms of search efficiency, the BC-RFMS scheme only presents the relationship between the size of its document set and the time required for searching, as illustrated in Figure 8.

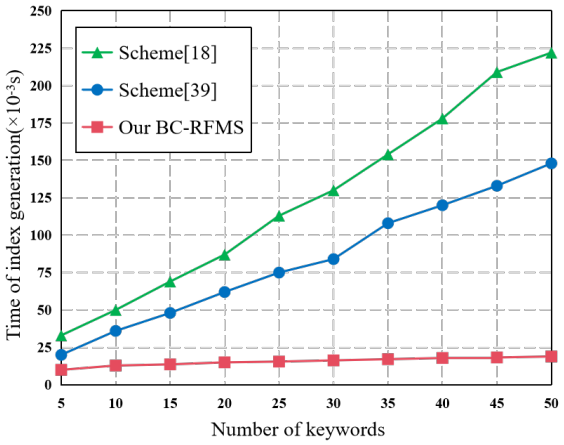


Figure 6. Time of index generation.

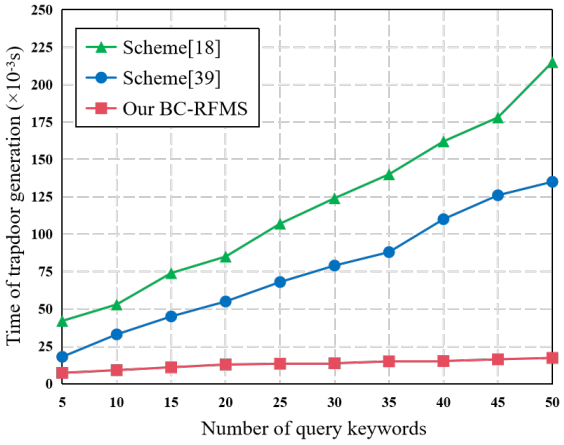


Figure 7. Time of trapdoor generation.

From the Figure 8, it is apparent that the search time in the BC-RFMS scheme is directly proportional to the number of documents. Additionally, we have conducted a comparison of the gas costs incurred by deploying smart contracts for searches, as shown in Table 2. Furthermore, we have evaluated the gas consumption for completing a transaction and the variation in gas costs for searching the same quantity of keywords against the size of the document set, as depicted respectively in Figures 9(a) and 9(b). Considering the blockchain

network's vulnerability to various consensus mechanisms, the operational performance on the blockchain can show variability, influencing gas consumption.

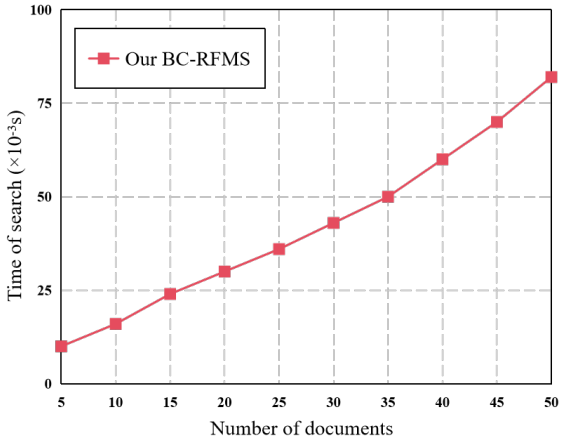
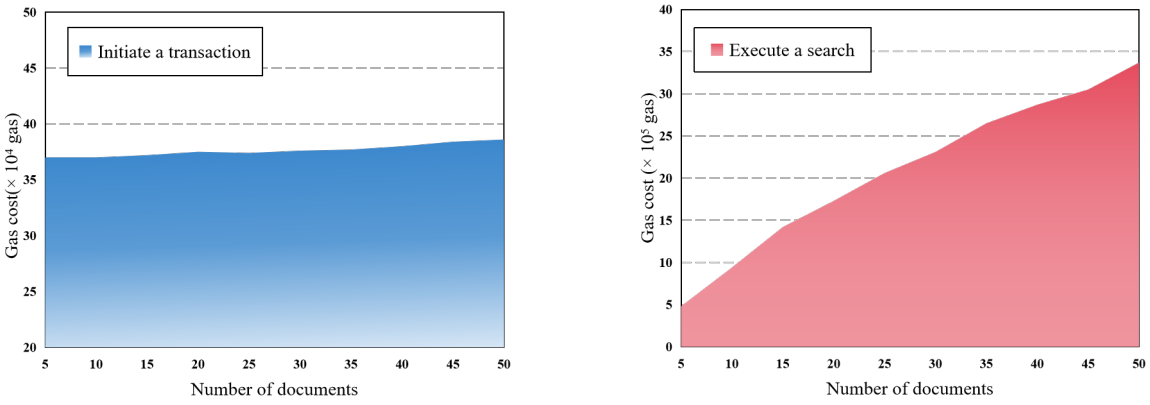


Figure 8. Time of search.

Table 2. Gas cost in test network.

Operation / Scheme	Deploying smart contracts
Chakraborty P. S. *et al*. [39]	13.6×10^5 gas
Our BC-RFMS	3.8×10^5 gas



(a) Per transaction

(b) Per search

Figure 9. Gas cost.

8. Conclusion and future work

In this paper, we proposed BC-RFMS, a blockchain-based rankable fuzzy multi-keyword search scheme. This scheme designed a smart contract for fuzzy search operations on the blockchain, ensuring the accuracy of search results. Additionally, we introduced a block-by-block transfer algorithm that segments index data for blockchain transmission, addressing the issue of transaction cessation due to exceeding the GasLimit in a single data transfer. Furthermore, our scheme stored encrypted files on the InterPlanetary File System (IPFS), achieving distributed storage and preventing document data loss due to single-point

storage failure. Experimental results demonstrate that the BC-RFMS scheme compared to existing solutions is secure, efficient and viable.

The scheme proposed in this paper effectively addresses the issue of Gas consumption during document transmission. However, it falls short when confronted with increasing data volumes. To achieve both scalability and efficiency improvement of the system, while ensuring high performance in handling increasing data volumes and maintaining system viability under various consensus mechanisms, will be the primary focus of our future research endeavors. Furthermore, we will explore a broader range of application scenarios, such as specific Internet of Things (IoT) contexts. With the IoT scenarios generating vast amounts of data, we will utilize this data to train our scheme model, thereby enhancing the optimization of our scheme.

Acknowledgments

This research is supported by National Natural Science Foundation of China (62102212), Shandong Provinceence Youth Innovation and Technology Program Innovation Team(2022KJ 296), Natural Science Foundation of Shandong (ZR202102190210), Nanchang Major Science and Technology Project (2023137), Science and Technology Small and Medium Enterprises (SMEs) Innovation Capacity Improvement Project of Shandong Province & Jinan City, China (Grant No. 2022TSGC2048) and Haiyou Famous Experts - Industry Leading TalentInnovation Team Project.

Conflicts of interests

The authors declared that they have no conflicts of interests.

Authors' contribution

Lixiang Zheng proposed the preliminary ideas of the paper. Lixiang Zheng, Hanlin Zhang and Xinrui Ge collaboratively discussed and designed the model definition, theoretical analysis and algorithm design of the proposed scheme. Lixiang Zheng conducted the experimental testing. Jie Lin, Fanyu Kong, and Leyun Yu participated in the related research work. Lixiang Zheng wrote the initial draft of the paper. The other authors assisted and revised the initial draft, leading to the final version of the paper. All authors contributed to the writing of the paper.

References

[1] Shen W, Yu J, Yang M, Hu J. Efficient Identity-Based Data Integrity Auditing with Key-Exposure Resistance for Cloud Storage. *IEEE Trans. Dependable Secure Comput.* 2023, 20(6):4593–4606.

[2] Kolhar M, Abu-Alhaj MM, Abd El-atty SM. Cloud data auditing techniques with a focus on privacy and security. *IEEE Secur. Priv.* 2017, 15(1):42–51.

[3] Zhang H, Gao P, Yu J, Lin J, Xiong NN. Machine learning on cloud with blockchain: a secure, verifiable and fair approach to outsource the linear regression. *IEEE Trans. Netw. Sci. Eng.* 2021, 9(6):3956–3967.

[4] Shen W, Gai C, Yu J, Su Y. Keyword-Based Remote Data Integrity Auditing Supporting Full Data Dynamics. *IEEE Trans. Serv. Comput.* 2023, pp. 1–4.

[5] Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE symposium on security and privacy.*, Berkeley, CA, United States, May 14–17, 2000, pp. 44–55.

[6] Goh EJ. Secure indexes. *Cryptology ePrint Archive* 2003 .

[7] Chang YC, Mitzenmacher M. In *Applied Cryptography and Network Security*, John Ioannidis MY Angelos Keromytis, ed., Berlin, Heidelberg: Springer, 2005, pp. 442–455.

[8] Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th ACM conference on Computer and communications security*, New York, United States, October 30, 2006, pp. 79–88.
[9] Wang C, Cao N, Li J, Ren K, Lou W. Secure ranked keyword search over encrypted cloud data. In *Proceedings of the 2010 IEEE 30th international conference on distributed computing systems*, New York, United States, June 21–25, 2010, pp. 253–262.
[10] Yin H, Qin Z, Ou L, Li K. A query privacy-enhanced and secure search scheme over encrypted data in cloud computing. *J. Comput. Syst. Sci.* 2017, 90:14–27.
[11] Golle P, Staddon J, Waters B. In *Applied Cryptography and Network Security*, Markus Jakobsson JZ Moti Yung, ed., Berlin, Heidelberg: Springer, 2004, pp. 31–45.
[12] Hwang YH, Lee PJ. In *Pairing-Based Cryptography - Pairing 2007*, Tsuyoshi Takagi EOTO Tatsuaki Okamoto, ed., Berlin, Heidelberg: Springer, 2007, pp. 2–22.
[13] Ballard L, Kamara S, Monrose F. In *Information and Communications Security*, Sihan Qing JLGW Wenbo Mao, ed., Berlin, Heidelberg: Springer, 2005, pp. 414–426.
[14] Boneh D, Waters B. In *Theory of Cryptography*, Vadhan SP, ed., Berlin, Heidelberg: Springer, 2007, pp. 535–554.
[15] Zhang B, Zhang F. An efficient public key encryption with conjunctive-subset keywords search. *J. Netw. Comput. Appl.* 2011, 34(1):262–267.
[16] Yao B, Li F, Xiao X. Secure nearest neighbor revisited. In *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering*, Brisbane, Australia, April 8–12, 2013, pp. 733–744.
[17] Li J, Wang Q, Wang C, Cao N, Ren K, *et al.* Fuzzy keyword search over encrypted data in cloud computing. In *2010 Proceedings IEEE INFOCOM*, San Diego, United States, March 14–19, 2010, pp. 1–5.
[18] Wang B, Yu S, Lou W, Hou YT. Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud. In *IEEE INFOCOM 2014-IEEE conference on computer communications*, Toronto, Canada, April 27–May 2, 2014, pp. 2112–2120.
[19] Wang J, Yu X, Zhao M. Privacy-preserving ranked multi-keyword fuzzy search on cloud encrypted data supporting range query. *Arab. J. Sci. Eng.* 2015, 40(8):2375–2388.
[20] Fu Z, Wu X, Guan C, Sun X, Ren K. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Trans. Inf. Forensics Secur.* 2016, 11(12):2706–2716.
[21] Zhong H, Li Z, Cui J, Sun Y, Liu L. Efficient dynamic multi-keyword fuzzy search over encrypted cloud data. *J. Netw. Comput. Appl.* 2020, 149:102469.
[22] Swaminathan A, Mao Y, Su GM, Gou H, Varna AL, *et al.* Confidentiality-preserving rank-ordered search. In *Proceedings of the 2007 ACM Workshop on Storage Security and Survivability*, New York, United States, October 29, 2007, pp. 7–12.
[23] Wang C, Cao N, Ren K, Lou W. Enabling secure and efficient ranked keyword search over outsourced cloud data. *IEEE Trans. Parallel Distrib. Syst.* 2011, 23(8):1467–1479.
[24] Cao N, Wang C, Li M, Ren K, Lou W. Privacy-preserving multi-keyword ranked search over encrypted cloud data. *IEEE Trans. Parallel Distrib. Syst.* 2013, 25(1):222–233.
[25] 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.
[26] Ding X, Liu P, Jin H. Privacy-preserving multi-keyword top-*k* k similarity search over encrypted data. *IEEE Trans. Dependable Secure Comput.* 2017, 16(2):344–357.
[27] Liu Q, Tian Y, Wu J, Peng T, Wang G. Enabling verifiable and dynamic ranked search over outsourced data. *IEEE Trans. Serv. Comput.* 2019, 15(1):69–82.
[28] Yin H, Qin Z, Zhang J, Ou L, Li F, *et al.* Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners. *Future Gener. Comput. Syst.* 2019, 100:689–700.

[29] Wang C, Ren K, Yu S, Urs KMR. Achieving usable and privacy-assured similarity search over outsourced cloud data. In *2012 Proceedings IEEE INFOCOM*, Orlando, United States, March 25–30, 2012, pp. 451–459.
[30] Kuzu M, Islam MS, Kantarcioglu M. Efficient similarity search over encrypted data. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, Arlington, United States, April 1–5, 2012, pp. 1156–1167.
[31] 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.
[32] Zhang H, Zhao S, Guo Z, Wen Q, Li W, *et al.* Scalable fuzzy keyword ranked search over encrypted data on hybrid clouds. *IEEE Trans. Cloud Comput.* 2021, 11(1):308 – 323.
[33] Li H, Zhang F, He J, Tian H. A searchable symmetric encryption scheme using blockchain. *arXiv* 2017, ArXiv:1711.01030.
[34] Hu S, Cai C, Wang Q, Wang C, Luo X, *et al.* Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization. In *Proceedings of the IEEE Conference on Computer Communications*, Honolulu, United States, April 16–19, 2018, pp. 792–800.
[35] Chen L, Lee WK, Chang CC, Choo KKR, Zhang N. Blockchain based searchable encryption for electronic health record sharing. *Future Gener. Comput. Syst.* 2019, 95:420–429.
[36] Li H, Tian H, Zhang F, He J. Blockchain-based searchable symmetric encryption scheme. *Comput. Electr. Eng.* 2019, 73:32–45.
[37] Zhang Y, Deng RH, Shu J, Yang K, Zheng D. TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain. *IEEE Access* 2018, 6:31077–31087.
[38] Yan X, Yuan X, Ye Q, Tang Y. Blockchain-based searchable encryption scheme with fair payment. *IEEE Access* 2020, 8:109687–109706.
[39] Chakraborty PS, Chandrawanshi MS, Kumar P, Tripathy S. BSMFS: Blockchain assisted Secure Multi-keyword Fuzzy Search over Encrypted Data. In *Proceedings of the 2022 IEEE International Conference on Blockchain*, Espoo, Finland, August 22–25, 2022, pp. 216–221.
[40] Xia Z, Wang X, Sun X, Wang Q. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Trans. Parallel Distrib. Syst.* 2015, 27(2):340–352.
[41] Fu S, Zhang Q, Jia N, Xu M. A privacy-preserving fuzzy search scheme supporting logic query over encrypted cloud data. *Mobile Netw. Appl.* 2021, 26:1574–1585.
[42] Qaiser S, Ali R. Text mining: use of TF-IDF to examine the relevance of words to documents. *Int. J. Comput. Appl.* 2018, 181(1):25–29.