Received 12 May 2023, accepted 6 June 2023, date of publication 9 June 2023, date of current version 28 June 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3284904*

RESEARCH ARTICLE

# A Rankable Boolean Searchable Encryption Scheme Supporting Dynamic Updates in a Cloud Environment

**SIXU GUO[1,2], HUIZHENG GENG[1], LI SU[1], SHEN HE[1], AND XINYUE ZHANG[1]**
[1]Department of Security Technology, China Mobile Research Institute, Beijing 100053, China
[2]Dalian Jiaotong University, Dalian 116021, China

Corresponding author: Huizheng Geng (genghuizheng@chinamobile.com)

This work was supported in part by Sixu Guo, Huizheng Geng, Li Su, Shen He, and Xinyue Zhang; and in part by the Beijing University of Posts and Telecommunications-China Mobile Research Institute Joint Innovation Center.

**ABSTRACT** At present, three problems exist in searchable encryption in cloud storage services: firstly, most traditional searchable encryption schemes only support single-keyword search while fail to perform Boolean searches; even if a few schemes support Boolean searching, the storage efficiency is also unsatisfactory. Secondly, most existing schemes do not support dynamic keyword updates, so the update efficiency is low. Thirdly, most existing schemes cannot meet all demands of users, to perform rankable searching over search files according to the importance of keywords. To solve these problems, a rankable Boolean searchable encryption scheme supporting dynamic updates in a cloud environment (RBDC) is proposed. By using Paillier and GM encryption algorithms, secure indices supporting dynamic updating are established. Based on applicable knowledge gleaned from cryptography and set theory, the indices of keyword intersections and the intersection search trapdoors are constructed to achieve multi-keyword Boolean search. With assistance of the SCP, score indices of each file are constructed according to the TF-IDF index, which allow ranking of files. Security analysis proofs that our scheme can ensure security in the known ciphertext model and the known background model. Experimental results prove that the scheme improves the search efficiency and the index storage efficiency.

**INDEX TERMS** Searchable encryption, Boolean search, rankable search, dynamic updating, privacy preserving.

## I. INTRODUCTION

### *A. BACKGROUND*

With the development of cloud computing, data are produced and transferred faster, heralding the age of big data. In the stage, with the explosive growth of data, increasingly more users and enterprises choose to outsource their data storage and business computation to a cloud server, which is entrusted to store and compute the data, thus saving data storage cost and reducing expenditure on system maintenance: however, just as a coin has two sides, cloud storage has not been popularized as expected. In recent years, increasingly frequent security accidents have been reported due to security threats to cloud service providers. To ensure confidentiality of data in a cloud environment, people prefer to encrypt data. Whereas, encrypted ciphertext data are meaningless, unrecognizable codes for everyone without the key. Likewise, after encrypting data, users also cannot use functions that seem very practical originally, taking a keyword search in cloud data as an example.

After much research, searchable encryption emerges as a technique [1], [2]. Searchable encryption refers to searching corresponding ciphertexts of non-structured information including files using keywords. The earliest concept of searchable encryption [1] was proposed by Goldreich and Ostrovsky. Traditional searchable encryption schemes

The associate editor coordinating the review of this manuscript and approving it for publication was Shu Xiao.

include two entities (clients and servers) and two stages (data initialization stage and search stage). In the data-initialization stage, a client generates an inverted index for each keyword and then encrypts the index to generate an encrypted index, followed by uploading the encrypted file set and encrypted index to the server. In the search stage, when a user initiates a search request, the client sends the search trapdoor of the keyword to the server. The trapdoor encapsulates the keyword based on the cryptography knowledge and cannot leak any key information. After it acquires the trapdoor, the server deciphers the encrypted index through methods including arithmetic operations and returns files that meet the search requirements.

### B. EXSISTING PROBLEMS

At present, most traditional searchable encryption schemes only support single-keyword search while fail to perform Boolean searches; even if a few schemes support Boolean searching, the storage efficiency is also unsatisfactory. Meanwhile, most existing schemes cannot meet all demands of users, to perform rankable searching over search files according to the importance of keywords. Therefore, the combination of rankable search and searchable encryption mechanisms is imperative (most existing searchable encryption schemes do not support rankable searching). When constructing searchable encryption schemes, three factors should be considered: privacy, efficiency, and query effectiveness [3], [4], [5], [6], [7], [8]. Although these factors are of equal importance, most existing schemes cannot balance them. Likewise, existing searchable encryption schemes mainly only support single-keyword search while fail to perform efficient multi-keyword Boolean searches. Even if there are a few schemes that support multi-keyword Boolean search, they are found to have extremely low storage efficiency and search efficiency [9], [10], [11], [12]. At the same time, most of traditional search schemes over ciphertexts do not support dynamic update [13], [14], [15] of keyword indices, this obviously cannot meet the actual needs of users. For example, for medical data, patient information is constantly changing,they are very inefficient when it becomes necessary to add, delete, or modify the keyword set to be searched. Therefore, it is necessary to construct a searchable encryption scheme that supports dynamic keyword updates, the invert search index is updated without the need to rebuild it, thus realizing keyword search, which greatly improves the efficiency of the scheme.

### C. RESEARCH CONTRIBUTIONS

Aiming at the aforementioned problems, a rankable Boolean searchable encryption scheme supporting dynamic updating in a cloud environment (RBDC) is proposed: this can meet daily demands of users and while performing secure and efficient multi-keyword Boolean search, the scheme ranks files according to user demands. The RBDC scheme uses set theory, which allows efficient multi-keyword Boolean searches. Meanwhile, forward file indices are constructed to rank searched files based on the secure coprocessor (SCP). Besides, the scheme constructs keyword indices using Goldwasser-Micali (GM) and Paillier encryption algorithms and enables dynamic index update. In the meantime, the use of elaborate structures of search trapdoors significantly improves the search efficiency and file ranking efficiency. Through security analysis, the scheme is proven to meet adaptive security demands.

Advantages of the research are demonstrated as follows: regarding functions, the RBDC scheme can rank searched files while supporting Boolean search compared with traditional searchable encryption schemes. Compared with the scheme in [4], the proposed scheme supports multi-keyword Boolean search. Compared with the scheme in [5] and [6], the scheme supports dynamic index updating. As for performance, the keyword search efficiency is first considered. Due to the special index construction of the RBDC scheme, the search efficiency is irrelevant to the index length, while it is only linearly related to the number of keywords. Therefore, the proposed scheme improves the Boolean search efficiency compared with the scheme in [5] and [6]. Considering the index storage efficiency, the index storage space in [6] is linearly correlated with the number of keywords. The scheme in [6] has been improved on this basis, leading to the sublinear correlation of the index storage space with the number of keywords. In comparison, because indices in the RBDC scheme are in the form of vectors, its index storage space is only related to the vector length while independent of the number of keywords. Hence, the index storage efficiency of the proposed scheme is of the order of its constants and increases substantially when there are many keywords.

The remainder of the paper is organized as follows: Section II introduces the theoretical knowledge needed for constructing the RBDC scheme. Section III introduces the model, formalized definition of algorithms, and security threats to the scheme. Section IV describes the design of the RBDC scheme. Section V demonstrates the security aspects and functions of the scheme and proves the superiorities of the scheme through comparison and experiments. Section VI draws conclusions.

## II. RELATED WORK

Since Goldreich proposed the concept of searchable encryption, the academic community has extensively studied the technique. Song et al. [16] developed a searchable encryption mechanism based on the symmetric encryption algorithm, and the scheme supports single-keyword search in the absence of indices. That is, the server scans the entirety of all documents to obtain search results. Since proposal of the mechanism, researchers have begun to attempt to use searchable encryption mechanisms in keyword search over ciphertexts. According to the construction of vector indices, Chang and Mitzenmacher [17] proposed their searchable encryption scheme based on a keyword dictionary established in advance. On this basis, Stefanov et al. [18] proposed a novel dynamic searchable encryption scheme, which leaks little information and has high search efficiency. In response to user demands, Yang et al. [19] proposed a searchable encryption method supporting query of multi-keywords in the union set and introduced knowledge of access authority, thus achieving control of the server over user authority. In the case of a large amount of user data, Cash et al. [20] developed a dynamic searchable encryption scheme, which optimizes secure indices by virtue of the multi-map structural characteristics and achieves single-keyword search over ciphertexts under conditions of large secure indices.

At present, the importance of searchable encryption has been reflected by its wide application fields and much research is underway. Searchable encryption has been applied to explore IT of Things (IoT) devices and intelligent electric meters [21]. It has also been proposed [22] that searchable encryption can be studied together with electronic health care systems in the cloud. Discussion in [23] shows that when combined with the blockchain technology, searchable encryption can also exert profound influences on secure trade. With the emergence of homomorphic encryption, researchers are also exploring to analyze and search human DNA sequences securely using searchable encryption in genome analysis [17].

Meanwhile, to ensure efficient and secure extraction of important information from big data and meet user demands, researchers also consider ranking various files according to some special keywords, so the concept of a rankable search emerges [24], [25], [26], [27], [28], [29], [30], [31]. Traditional ranking search schemes have two parameters, $(k, w)$, where parameter $k$ controls the size of the returned result set, and parameter $w$ represents the user's preferences on various attributes of the data, which may be a weight vector or a score value, such as TF-IDF score. Fagin et al. [24] took the lead to propose the concept of rankable search, aiming to solve document retrieval problems when searching big data; because of the specific requirements for ranking of data including files in many scenarios, rankable searching has attracted much attention since its proposal and has been widely studied and applied in search engine (SE), e-commerce, and mobile applications (apps). Users show their preference by setting weights to different attributes of keywords, while Cloud Server uses weights provided by users as the ranking basis in computation and then returns ranked data according to user demand. Rankable searches help users to find information of interest from huge datasets, justifying the present study.

The ranked keyword search solutions have been extensively studied. In addition to some conventional ranking search algorithms, such as TA [24], NRA [26], Upper & Pick [27], many new technologies related to keyword rankable search have also been proposed. The most representative is the Reverse Ranking algorithm proposed by Valchou et al [31]. As the amount of data increases, distributed rankable search is receiving increasing attention. [28] proposed rankable search for dispersed web databases in the network. Reference [30] proposed a method called SPEERTO for distributed ranking search. At present, new technologies for keyword rankable search are increasingly being studied. Rankable search has been applied to explore industrial IoT devices with efficient key management [32]. With the development of machine learning, Miao et al proposed a ranked keyword search through machine learning protocol in 2023 [33].

## III. PRELIMINARY

### A. GM PUBLIC-KEY ENCRYPTION SCHEME

The GM scheme [34] is the first public-key encryption scheme proven secure under the standard cryptographic assumption. It consists of three algorithms.

#### 1) GEN

The key generation algorithm. A public key $pk = (n, m)$ is selected, in which $n = p \cdot q$, and $p, q$ are two large odd primes. Meanwhile, $m$ meets the Jacobi symbol $J\left(\frac{m}{n}\right) = 1$. A private key $sk = (p, q)$ is selected and $\mathcal{R} \in \mathbb{Z}_n^*$ is chosen as the set of random numbers.

#### 2) ENC

The encryption algorithm. A plaintex $M$ is transformed into a binary digit $x_i \in (0, 1)$. By using the public key $pk$ and a random number $r_i \in \mathcal{R}$, $c_i = m^{x_i} r_i^2 \bmod n$ is calculated for each bit $x_i$, thus determining the ciphertext set $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$.

#### 3) DEC

The deciphering algorithm. For a ciphertext $c_i \in \mathcal{C}$, the private key $sk$ is adopted to decipher each bit $c_i$ in the ciphertext, thus attaining the plaintext $x_i$:

$$x_i = \begin{cases} 0, J\left(\dfrac{c_i}{p}\right) = 1, J\left(\dfrac{c_i}{q}\right) = 1. \\ 1, J\left(\dfrac{c_i}{p}\right) = -1, J\left(\dfrac{c_i}{q}\right) = -1. \end{cases} \tag{1}$$

As GM encryption is executed using a probabilistic algorithm, different ciphertexts are produced upon each encryption of the given plaintext, bestowing significant advantages.

### B. PAILLIER HOMOMORPHIC ENCRYPTION SCHEME

The Paillier encryption scheme [35] consists of three algorithms:

#### 1) GEN

A public key $pk = (n, g)$ is selected, where $n = pq$, gcd $(pq, (p-1)(q-1)) = 1$. Next, the following is calculated

$$\lambda = lcm\,(p-1, q-1)\,, \mu = \left(\frac{g^{\lambda} \bmod n^2 - 1}{n}\right)^{-1} \bmod n \tag{2}$$

and the private key $sk = (\lambda, \mu)$ is obtained.

2) ENC

For a plaintext $m < n$, the ciphertext is calculated as

3) DEC

For a ciphertext $c < n^2$, the plaintext is deduced as

$$c = g^m \cdot r^n \bmod n^2 \tag{3}$$

*Additively Homomorphism [36]:*

$$Enc\,(a_1 + a_2) = Enc\,(a_1)\, Enc\,(a_2) \tag{4}$$

### C. TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY

Term frequency-inverse document frequency (TF-IDF) [37], as a common information retrieval technique, is mainly adopted to weight common keywords in documents. TF-IDF is commonly used as the statistical method in information retrieval, to evaluate the importance of a term for a file in a file set or a text corpus, making it perfect for rankable searches. The technique not only pays attention to the term frequency (TF) because some words such as "this" and "the" are of low significance despite of their high frequency of occurrence in articles. Therefore, the importance of terms in TF-IDF not only increases in direct proportion to their occurrence times in files but also decreases in inverse proportion to their frequency of occurrence in the text corpus, avoiding the problem.

### D. BOOLEAN SEARCH AND SET THEORY

In the process of Boolean search, the generated logical searches need to experience disjunction at first, thus obtaining a standard keyword connection $\delta_1 \wedge \cdots \wedge \delta_\ell$. Therein, for any $\delta_i$, $\delta_i = w_{i,1} \vee \cdots \vee w_{i,q}$. For the intersection of sets, it is necessary to find only that content shared between the two sets; for the union set, it is necessary to search for the intersection of sets and then subtract the intersection from their sum (an onerous process).

In the knowledge system of set theory, a method is available to transform union operation into the intersection operation [19]. For example, there are three sets $\mathcal{D}(w_1), \mathcal{D}(w_2), \mathcal{D}(w_3)$ with the relationship in Figure 1.

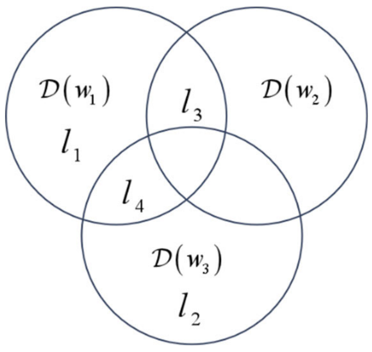

**FIGURE 1. Set relationship diagram.**

The deduction process for the union operation of the three sets is expressed as follows:

$$\begin{aligned}
&\mathcal{D}(w_1) \vee \mathcal{D}(w_2) \vee \mathcal{D}(w_3) \\
&\quad = (l_1, l_2, l_3, l_4) \leftrightarrow (l_1) + (l_3) + (l_2, l_4) \\
&\quad = \mathcal{D}(w_1) - (\mathcal{D}(w_1) \wedge \mathcal{D}(w_2)) - (\mathcal{D}(w_1) \wedge \mathcal{D}(w_3)) \\
&\qquad + \mathcal{D}(w_2) - (\mathcal{D}(w_2) \wedge \mathcal{D}(w_3)) + \mathcal{D}(w_3) \\
&\quad = (l_1, l_3, l_4) - (l_3) - (l_4) + (l_3) - (\emptyset) + (l_2, l_4).
\end{aligned} \tag{5}$$

In this way, the deduction process of replacing the union operation with the intersection operation is fulfilled.

## IV. THE RBDC SCHEME

The section describes the model, formalized definition, and security threats of the RBDC scheme.

### A. RBDC MODEL

The RBDC scheme involves three entities: the Data Owner, Cloud Server, and SCP, as displayed in Figure 2.

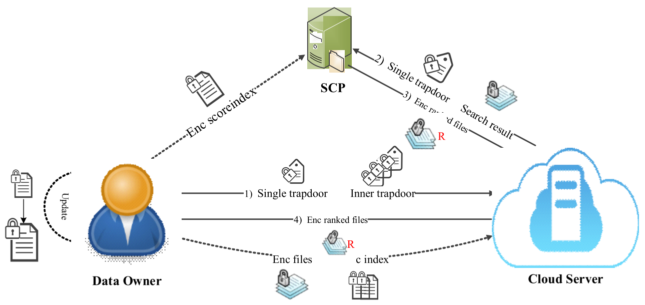

**FIGURE 2. The architecture of the RBDC scheme.**

For the model of the RBDC scheme in Figure 2, the off-line transmission phase indicated by dotted lines in the figure is executed first. The Data Owner transfers the encrypted files and encrypted keyword indices to a Cloud Server, and transfers encrypted score indices to the SCP [34]. The on-line transmission phase is then executed by the RBDC scheme, including four steps:

1) The Data Owner transfers the search trapdoor to a Cloud Server and initiates the Search Request;
2) After searching, the Cloud Server transfers the search results and search trapdoors to the SCP;
3) After ranking search results by score indices, the SCP transfers the ranked encrypted files to a Cloud Server;
4) The Cloud Server transfers the ranked encrypted files to the Data Owner, fulfilling the RBDC scheme.

Besides, when the keyword set changes, the RBDC scheme supports dynamic updating of keyword indices.

In the model, SCP is regarded as a small proxy server that resides in the isolated execution environment provided by a Cloud Server and is considered trustworthy. The Data Owner firstly encrypts files then uploads them to a Cloud Server. A Cloud Server is curious but honest. Detailed introductions to the functions of each entity in the scheme are provided below:

#### 1) DATA OWNER

It is responsible for generating the keys used in the entire search method and generating encrypted files using the symmetric encryption algorithm, that is, it is responsible for the initiation. It is required to generate the keyword sets corresponding to their file sets and produce inverted indices according to the two. The Data Owner also needs to process the generated inverted indices, which are encrypted and then uploaded to a Cloud Server, that is, in charge of generation of keyword indices. To fulfil a Boolean search, the indices include two parts: indices of single keywords and indices of keyword intersections. To achieve a rankable search, the Data Owner has to generate a forward score index for each file. Meanwhile, the Data Owner also should generate trapdoors corresponding to each keyword to be searched. Meanwhile, when returning the search results, the Data Owner needs to decipher the data to obtain the searched files. If the keyword set varies (being added, deleted, or modified), it becomes necessary to update the keyword set dynamically.

#### 2) CLOUD SERVER

A Cloud Server mainly receives encrypted indices transferred from the Data Owner. It also receives Search Requests from trusted users and corresponding search trapdoors and performs search. Meanwhile, a Cloud Server also receives Search Requests sent by the Data Owner and the file set ranked by SCP.

#### 3) SCP

SCP is mainly in charge of receiving encrypted score indices in the scheme, receiving encrypted character strings transferred by a Cloud Server to search and rank keywords, and returning corresponding document tags thereto.

### B. FORMALIZED DEFINITION

The RBDC scheme is expressed as follows:

$$\text{RBDC}=\left(\begin{array}{l} KeyGen, IndexGen, ScoreIndexGen, \\ TrapdoorGen, Search, Rank, Update \end{array}\right). \quad (6)$$

Each algorithm is described as follows:

Key generation algorithm MK, $K_{\text{PL}}$, $K_{\text{GM}} \leftarrow KeyGen(\lambda)$ is a probabilistic algorithm that is run by the Data Owner. The security parameter $\lambda$ is input while the symmetric encryption key MK, public and private keys $K_{\text{PL}} = (pk_{\text{PL}}, sk_{\text{PL}})$ for Paillier encryption, and keys $K_{\text{GM}} = (pk_{\text{GM}}, sk_{\text{GM}})$ for GM encryption are output.

The generation algorithm of keyword indices $\boldsymbol{EIndex} \leftarrow IndexGen(pk_{\text{PL}}, p\,k_{\text{GM}}, \mathcal{V}, \mathcal{W}, \mathcal{R})$ is a probabilistic algorithm that is run by the Data Owner. The keys $pk_{\text{DN}}$, $p\,k_{\text{GM}}$, a set $\mathcal{V}$ of random orthogonal vectors, a keyword set $\mathcal{W}$, and a set $\mathcal{R}$ of random numbers are input, while the encrypted keyword index $\boldsymbol{EIndex}$ is output.

The generation algorithm of encrypted score indices $\boldsymbol{EIndex}_{\text{Score}} \leftarrow ScoreIndexGen(pk_{\text{PL}}, \mathcal{D}, \mathcal{V}, \mathcal{W}, \mathcal{R})$ is a probabilistic algorithm that is run by the Data Owner. The key $pk_{\text{DN}}$, a set $\mathcal{V}$ of random orthogonal vectors, a document set $\mathcal{D}$, a keyword set $\mathcal{W}$, and a set $\mathcal{R}$ of random numbers are input, while the encrypted score index $\boldsymbol{EIndex}_{\text{Score}}$ is output.

Trapdoor generation algorithm $t_{w_q} \leftarrow TrapdoorGen(K_{\text{pL}}, \mathcal{W}, \mathcal{V})$ is a deterministic algorithm that is run by the Data Owner. The public key $pk_{\text{pL}}$ for Paillier encryption, a set $\mathcal{V}$ of random orthogonal vectors, and a keyword set $\mathcal{W}$ are input while the keyword search trapdoor $t_{w_q}$ is output.

Search algorithm $m_{w_q} \leftarrow Search(\boldsymbol{EIndex}, t_{w_q})$ is a deterministic algorithm that is run by a Cloud Server. The encrypted secure index $\boldsymbol{EIndex}$ and search trapdoor $t_{w_q}$ are input while the encrypted character string $m_{w_q}$ corresponding to each keyword searched is output.

Ranking algorithm $\mathcal{D}_r \leftarrow Rank(m_{w_q}, s\,k_{\text{GM}}, \boldsymbol{EIndex}_{\text{Score}}, t_{w_q})$ is a deterministic algorithm that is run by SCP. The encrypted character string $m_{w_q}$, key $sk_{\text{GM}}$, search trapdoor $t_{w_q}$, and encrypted score index $\boldsymbol{EIndex}_{\text{Score}}$ are input while the ranked document set $\mathcal{D}_r$ is output.

Update algorithm $\boldsymbol{EIndex}' \leftarrow Update(\boldsymbol{EIndex}, w')$ is a probabilistic algorithm that is run by the Data Owner. The encrypted index $\boldsymbol{EIndex}$ and keyword $w'$ to be updated are input while the new encrypted index $\boldsymbol{EIndex}'$ is output.

### C. SECURITY THREATS

Security threats in two models, namely, the known ciphertext model and the known background model are considered in the RBDC scheme according to previous research [19], [27]. The Data Owner and SCP are considered as completely trusted entities in the scheme, while any Cloud Server is curious but honest. The Cloud Server honestly stores all data documents belonging to the Data Owner according to the specified protocol of algorithms while it is curious about the stored data. That is, the Cloud Server tends to obtain all data and metadata pertaining to the Data Owner by inferring or analyzing enciphered data and search trapdoors.

#### 1) THE KNOWN CIPHERTEXT MODEL

In the known ciphertext model, it is assumed that a Cloud Server can only obtain encrypted document set encrypted document set $\mathcal{C}$, encrypted index $\boldsymbol{Elndex}$, and search trapdoor $t_{w_q}$ uploaded by the Data Owner as well as mastered by the Cloud Server. Whereas, it fails to acquire other information according to the relationships between various indices and between trapdoors; because SCP is regarded as completely trusted, information of the encrypted score indices and the entire ranking process are not considered. Therefore, the security threat is expressed as $\left\langle \boldsymbol{EIndex}, t_{w_q}, ID_i, \tilde{\boldsymbol{Elndex}} \right\rangle$ in the known ciphertext model. Meanwhile, because the scheme supports the dynamic update of keyword indices, forward security before and after index update needs to be taken into account, that is, the Cloud Server cannot obtain relevant information of the updated index $\boldsymbol{EIndex}'$ from the index $\boldsymbol{EIndex}$ before updating.

#### 2) THE KNOWN BACKGROUND MODEL

In the known background model, a Cloud Server can acquire more data and information than that in the known ciphertext model, including information between keyword indices, between search trapdoors, and statistical information between data sets. Hence, a Cloud Server has stronger attack ability. A Cloud Server can infer and analyze uploaded encrypted indices, search trapdoors, and search results according to known trapdoor information and some statistical information, thus determining plaintext information of some keywords in the search.

## V. DETAILED DESIGN OF THE RBDC SCHEME

The detailed design of the RBDC scheme is introduced herein: the RBDC scheme must be able to address the following three problems:

1) to achieve efficient multi-keyword Boolean searching;
2) to rank searched files;
3) to update the constructed secure indices of keywords dynamically.

In accordance with the formalized definition in Section II, the seven algorithms in the RBDC scheme are described below.

### A. THE KEY GENERATION ALGORITHM

The generation algorithm MK, $K_{\mathrm{PL}}, K_{\mathrm{GM}} \leftarrow KeyGen(\lambda)$ is realized by the Data Owner. The RBDC scheme encrypts files using the traditional symmetric encryption modes (such as AES), inputs the security parameter $\lambda$, and generates the file encryption key MK. The GM and Paillier encryption algorithms are used for constructing the encrypted indices and search trapdoors. The security parameter $\lambda$ is input and public and private keys $K_{\mathrm{PL}} = \left(sk_{\mathrm{PL}}, pk_{\mathrm{PL}}\right)$, $K_{\mathrm{GM}} = (sk_{\mathrm{GM}}, pk_{\mathrm{GM}})$ for the two encryption algorithms are generated. According to relevant knowledge:

$$\begin{aligned} pk_{\mathrm{PL}} &= (\lambda, \mu), \quad & sk_{\mathrm{PL}} &= (\lambda, \mu) \\ pk_{\mathrm{GM}} &= (n, m), \quad & sk_{\mathrm{GM}} &= p \end{aligned} \tag{7}$$

### B. THE GENERATION ALGORITHM OF KEYWORD INDICES

The generation algorithm of keyword indices ***EIndex*** $\leftarrow$ $IndexGen\left(pk_{\mathrm{PL}}, p\,k_{\mathrm{GM}}, \mathcal{V}, \mathcal{W}, \mathcal{R}\right)$ is realized by the Data Owner. To achieve Boolean search, knowledge in the set theory in prerequisite knowledge is adopted. The encrypted keyword indices constructed in the scheme consist of two parts: encrypted indices ***sEIndex*** for single keywords and encrypted indices ***iEIndex*** for keyword intersections.

#### 1) GENERATION OF ENCRYPTED INDICES FOR SINGLE KEYWORDS

At first, the Data Owner generates a binary index string $biw_i$ with the length of $|\mathcal{D}|$ for each keyword $w_i \in \mathcal{W}$. That is to say, if the file $d_j \in \mathcal{D}$ contains the keyword $w_i$, the $j$ th bit of $biw\ w_i$ is 1; otherwise, it is 0. Each $biw_i$ is stored in the data structure of a dictionary and recorded as $biD\,(w_i)$, with the size of $|\mathcal{W}|$. The construction of $biD\ (w_i)$ is shown in Figure 3.

For each element in the dictionary, $G_i = Enc_{\mathrm{GM}}\,(pk_{\mathrm{GM}}, bID\,(w_i))$ is generated through GM encryption. $\mathcal{V}$ is defined

$$biD\left(w_i\right) = \begin{cases} w_1 \rightarrow f_1, f_3, f_5 \rightarrow 10101\ldots \\ w_2 \rightarrow f_2, f_3, f_4 \rightarrow 01110\ldots \\ w_3 \rightarrow f_1, f_2, f_3 \rightarrow 11100\ldots \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \end{cases}$$

**FIGURE 3. Construction of a binary dictionary.**

as a mutually orthogonal vector set; $\boldsymbol{v}_i \in \mathcal{V}$ is a random vector corresponding to each keyword; $r \in \mathcal{R}$ is a random number; $\boldsymbol{v}_r \in \mathcal{V}$ is a random vector. For each keyword $w_i, P_i = Enc_{\mathrm{PL}}\,(pk_{\mathrm{PL}}, w_i)$ is generated via Paillier encryption. The encrypted index vector of single keywords corresponding to the keyword set $\mathcal{W}$ generated using the aforementioned steps is defined as ***sEIndex***, for which:

$$\boldsymbol{sEIndex} = \sum_{i=1}^{|\mathcal{W}|} (\boldsymbol{v}_i \cdot G_i \cdot P_i) + \boldsymbol{v}_r \cdot r. \tag{8}$$

#### 2) GENERATION OF ENCRYPTED INDICES OF KEYWORD INTERSECTIONS

Based on relevant knowledge of the set theory, the Data Owner first ascertains the intersection between each keyword $w_i \in \mathcal{W}$ and keyword $w_j \in \mathcal{W}$ behind, thus generating the inverted index $inID_i$ of $(|\mathcal{W}\|\mathcal{D}|)$ intersections. Similar to generation of encrypted indices of single keywords, a binary index string $inbID_i$ with the length of $|\mathcal{D}|$ is generated according to the inverted index of each keyword intersection and then stored in a dictionary. Each dictionary is placed in a multi-map $MMb\,(w_i)$. For each element in $MMb\,(w_i)$, the method akin to that used to generate encrypted indices of single keywords is adopted to generate encrypted indices. $inG_{i\cap j} = Enc_{\mathrm{GM}}\,(pk_{\mathrm{GM}}, i\ nbID_i)$ is generated through GM encryption. XOR operation is performed for keywords undergoing the intersection operation, for which $inP_{i\cap j} = Enc_{\mathrm{PL}}\left(pk_{\mathrm{PL}}, w_i \oplus w_j\right)$ is generated through Paillier encryption. Other operations resemble those for generating encrypted indices of single keywords. The Data Owner places the encrypted index vectors of intersections corresponding to each keyword into the dictionary $inD\,(w_i)$, as given by

$$inD\,(w_i) = \sum_{j=1+i}^{|\boldsymbol{\mathcal{W}}|} \left(\boldsymbol{v}_i \cdot inG_{i\cap j} \cdot inP_{i\cap j}\right) + \boldsymbol{v}_r \cdot r. \tag{9}$$

***iEIndex*** represents all elements in the dictionary $inD(w_i)$. In this way, the index ***iEIndex*** of keyword intersections is generated.

### C. THE GENERATION ALGORITHM OF ENCRYPTED SCORE INDICES

The generation algorithm of encrypted score indices $\boldsymbol{EIndex}_{\mathrm{Score}} \leftarrow ScoreIndexGen\left(pk_{\mathrm{PL}}, \mathcal{D}, \mathcal{V}, \mathcal{W}, \mathcal{R}\right)$ is realized by the Data Owner. Score indices are forward indices, that is, each file $d_j$ corresponds to a string of indices, which is different from keyword indices. The Data Owner firstly calculates the TF-IDF value of the keyword $w_i$ in files to construct the score indices, providing convenience for the subsequent file ranking. After calculating the TF-IDF value of files, it is stored in the dictionary $sD(d_j)$. The algorithm is expressed as follows:

$$sD\left(d_j\right) = \sum_{i=1}^{w_i \in d_j} \left(tf\_idf\left(w_i, d_j\right)\right). \tag{10}$$

Akin to the method of encrypting keyword indices, the Data Owner leverages Paillier encryption and random vectors to encrypt each string of score indices in Data Owner leverages Paillier encryption and random vectors to encrypt each string of score indices in $sD\left(d_j\right)$ to generate $D_i$. Meanwhile, $\boldsymbol{v}_i$ that has been generated when calculating encrypted keyword indices is fetched to generate the encrypted score index $\boldsymbol{EIndex}_{\text{score}}$ according to the TF-IDF value in the dictionary, the calculated value of $P_i$, and $\boldsymbol{v}'_r$.

$$\boldsymbol{EIndex}_{\text{Score}} = \sum_{i=1}^{w_i \in d_j} \left(\boldsymbol{v}_i \cdot P_i \cdot tf\_idf\left(w_i, d_j\right)\right) + \boldsymbol{v'}_r \cdot r'. \tag{11}$$

### D. THE TRAPDOOR GENERATION ALGORITHM

$t_{w_q} \leftarrow TrapdoorGen\,(K_{\text{PL}}, \mathcal{W}, \mathcal{V})$ is realized by the Data Owner to facilitate search and ranking procedures. To realize Boolean searching, the trapdoor $t_{w_q} = \left(st_{w_q}, i\,t_{w_q}\right)$ corresponding to each keyword $w_q$ in RBDC comprises two parts: trapdoor $st_{w_q}$ for single keywords and the trapdoor $it_{w_q}$ for keyword intersections.

#### 1) GENERATION OF THE TRAPDOOR FOR SINGLE KEYWORDS

For each keyword For each keyword $w_q$, the trapdoor $st_{w_q}$ for single keywords is generated at first. For each keyword $w_q \in \mathcal{W}$, the Data Owner firstly takes the key $pk_{\text{pL}} = (n, g)$ to generate $P_{w_q}^{-1} = Enc_{\text{pL}}\left(pk_{\text{pL}}, -w_q, r_q\right)$, in which $r_q$ must satisfy the following formula:

$$r_q \cdot r_{w_q} = 1 \bmod n^2, \quad r_q \in Z_n^* \tag{12}$$

where $r_{w_q}$ is the random number used in Paillier encryption corresponding to the keyword $w_q$ when generating encrypted indices. $\boldsymbol{v}_{w_q} \in \mathcal{V}$ is the random vector corresponding to the keyword $w_q$. The formula for generating the trapdoor $st_{w_q}$ for single keywords is as expressed by Eq. (13):

$$st_{w_q} = \boldsymbol{v}_{w_q} \cdot P_{w_q}^{-1} \tag{13}$$

#### 2) GENERATION OF THE TRAPDOOR FOR KEYWORD INTERSECTIONS

The Data Owner then generates the intersection search trapdoor for each keyword. The intersection search trapdoor $it_{w_q}$ is composed of two parts: because the last keyword does not need to be subjected to the intersection operation with other keywords, the search trapdoors generated for the first $q-1$ keywords and that for the last keyword have different structures. In the set $\mathcal{W}_{q-1}$ of the first $q-1$ keywords, $P_{w_q \cap w_i}^{-1} = Enc_{\text{pL}}\left(pk_{\text{pL}}, -w_{q\cap i}, r_{q\cap i}\right)$ is generated for each keyword $w_q \in \mathcal{W}_{q-1}$ and the keyword $w_i \in \mathcal{W}_{q-1}$ behind in a way similar to that for generating trapdoors for single keywords. Therein, $r_{q\cap i}$ should satisfy the following formula:

$$r_{q\cap i} \cdot r_{w_q \cap w_i} = 1 \bmod n^2, \quad r_{q\cap i} \in Z_n^* \tag{14}$$

$r_{q\cap i}$ is the random number used in Paillier encryption corresponding to the keyword $w_q \cap w_i$ when generating encrypted indices of intersections. Other steps are similar to those for generating the trapdoor $st_{w_q}$ for single keywords. The construction of the trapdoor for keyword intersections is governed by

$$it_{w_q \cap w_i} = \left(\boldsymbol{v}_{w_q} \cdot P_{w_q \cap w_i}^{-1}\right). \tag{15}$$

Each $it_{w_q \cap w_i}$ is integrated to generate the search trapdoor $it_{w_q}$ for intersections corresponding to $w_q$. For the last keyword $w_q$, it does not need to have the intersection operation with other keywords but only its single-keyword trapdoor $st_{w_q}$ is needed. The two parts of trapdoors are integrated to attain $t_{v_q}$, the construction of which is displayed in Figure 4.

$$\begin{aligned}
t_{w_1} &= \left(st_{w_1}, it_{w_1 \cap w_2}, \ldots,\ it_{w_1 \cap w_q}\right),\\
t_{w_2} &= \left(st_{w_2}, it_{w_2 \cap w_3}, \ldots,\ it_{w_2 \cap w_q}\right),\\
&\vdots\\
t_{w_{q-1}} &= \left(st_{w_{q-1}}, it_{w_{q-1} \cap w_q}\right),\\
t_{w_q} &= st_{w_q}.
\end{aligned}$$

**FIGURE 4. Construction of keyword trapdoors.**

### E. THE SEARCH ALGORITHM

The search algorithm $m_{w_q} \leftarrow Search\left(\boldsymbol{EIndex}, t_{w_q}\right)$ is realized by a Cloud Server. The Data Owner sends the keyword set realized by a Cloud Server. The Data Owner sends the keyword set $\mathcal{W}'$ to be searched and the Search Request to a Cloud Server, in which the number of keywords is $\left|\mathcal{W}'\right|$. A Cloud Server traverses search trapdoors of the first $\left(\left|\mathcal{W}'\right| - 1\right)$ keywords. For each trapdoor $t_{w_q} = \left(st_{w_q}, it_{w_q}\right)$, $s_{w_q}$ is fetched to search $\boldsymbol{sEIndex}$, thus obtaining $sm_{w_q}$. The deduction process is shown as follows:

$$sm_{w_q} = \left(\boldsymbol{sEIndex} \cdot st_{w_q}\right) \bmod n^2 = G_{w_q}. \tag{16}$$

When $w_q \in \mathcal{W}$, the parameter $sm_{w_q}$ obtained by a Cloud Server is the binary index string $G_{w_q}$ corresponding to $w_q$ encrypted by GM. Each searched $sm_{w_q}$ is placed in the dictionary $D_{sm}\left(w_q\right)$ (Figure 5).

$$D_{sm}\left(w_q\right)=\begin{cases} w_1 \to \boldsymbol{sEIndex} \xrightarrow{stw_1} sm_{w_1} \\ w_2 \to \boldsymbol{sEIndex} \xrightarrow{stw_2} sm_{w_2} \\ \vdots \qquad \vdots \qquad \vdots \\ w_q \to \boldsymbol{sEIndex} \xrightarrow{stw_q} sm_{w_q} \end{cases}$$

**FIGURE 5. Construction of single-keyword search results.**

Then, a Cloud Server takes $it_{w_q}$ to search ***iEIndex*** and fetches index vectors corresponding to each keyword from the dictionary $inD\left(w_i\right)$. The method is akin to the search method of ***sEIndex***. The resulting parameter $inmw_{q\cap i}$ is the binary index string $inG_{q\cap i}$ attained by GM encryption of $w_q$ and the keyword $w_i$ behind. Each searched parameter $inm_{w_{q\cap i}}$ is placed in multi-map $MM_{inm}\left(w_q\right)$ (Figure 6).

$$MM_{inm}\left(w_q\right)=\begin{cases} w_1\cap\begin{cases} w_2 \to inD\left(w_1\right)\xrightarrow{it_{w_1\cap w_2}} inm_{w_1\cap w_2} \\ w_3 \to inD\left(w_1\right)\xrightarrow{it_{w_1\cap w_3}} inm_{w_1\cap w_3} \\ \vdots \qquad \vdots \qquad \vdots \\ w_q \to inD\left(w_1\right)\xrightarrow{it_{w_1\cap w_q}} inm_{w_1\cap w_q} \end{cases} \\ w_2\cap\begin{cases} w_3 \to inD\left(w_2\right)\xrightarrow{it_{w_2\cap w_3}} inm_{w_2\cap w_3} \\ \vdots \qquad \vdots \qquad \vdots \\ w_q \to inD\left(w_2\right)\xrightarrow{it_{w_2\cap w_q}} inm_{w_2\cap w_q} \end{cases} \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ w_{q-1}\cap w_q \to inD\left(w_{q-1}\right)\xrightarrow{it_{w_{q-1}\cap w_q}} inm_{w_{q-1}\cap w_q} \end{cases}$$

**FIGURE 6. Construction of multi-keyword search results.**

For the last keyword $w_q$, only its single-keyword trapdoor $s\tau_{w_q}$ needs to be taken to search ***SEIndex***, thus obtaining $sm_{w_q} = G_{w_q}$. After placing $sm_{w_q}$ in the dictionary $D_{sm}\left(w_q\right)$, the search process is fulfilled.

## F. THE RANKING ALGORITHM

Ranking algorithm $\mathcal{D}_r \leftarrow Rank\left(m_{w_q}, sk_{\text{GM}}, \boldsymbol{EIndex}_{\text{Score}}, t_{w_q}\right)$ is achieved by SCP and can be used to rank documents and return the ranked file set $\mathcal{D}_r$. After executing the search algorithm, a Cloud Server sends $m_{w_q}$ to SCP. SCP firstly uses $sk_{\text{GM}}$ to decipher $sm_{w_q}$ and $it_{w_q}$, thereby attaining the binary index string $biw_q$ of each keyword $w_q$, and the binary index string ${}_{in}biw_{q\cap r}$ for the intersection of $w_q$ and each keyword $w_i$ behind. For all keywords $\mathcal{W}'$ involved in a Boolean search, the following operation is performed for all keywords and $\mathcal{W}$ executing the union operation:

$$_{and}bi = \sum_{w_q\in and\mathcal{W}} biw_i - \sum_{w_q\in and\mathcal{W}} \left( \sum_{w_i\in and\mathcal{W}}^{i>q} {}_{in}biw_{q\cap i} \right) \tag{17}$$

By doing so, $_{and}bi$ can be determined. Here, $\Sigma$ refers to bit-by-bit addition. Thereafter, the following operation is performed for $_{and}bi$ with ${}_{in}biw_{q\cap r}$ of all keywords $in\mathcal{W}$ executing the intersection operation:

$$bi\mathcal{D}' = {}_{and}bi \cap \left( \bigcap_{w_q\in in\mathcal{W}}^{i>q} {}_{in}biw_{q\cap i} \right) \tag{18}$$

In this way, the binary index string $bi\mathcal{D}'$ of all file sets for Boolean search is obtained, and then the set $\mathcal{D}'$ of all files searched is acauired.

SCP uses score indices to achieve ranking and adopts the trapdoor $st_{w_q}$ for single keywords to decipher $\boldsymbol{EIndex}_{\text{score}}$. The ranking score dictionary $sD\left(d_j\right)$ that stores $d_j$ is then obtained. SCP ranks scores of all files in $sD\left(d_j\right)$ and returns the ranked file set $\mathcal{D}_r$ to a Cloud Server. In this way, the entire ranking algorithm is fulfilled.

## G. THE DYNAMIC UPDATING ALGORITHM

Dynamic algorithm $\boldsymbol{EIndex}' \leftarrow Update\left(\boldsymbol{EIndex}, w'\right)$ is achieved by the Data Owner. While updating the keyword set $\mathcal{W}$, the RBDC scheme dynamically updates the encrypted keyword indices ***EIndex***, thus greatly improving the updating efficiency of keyword indices. The index updating algorithm involves the updating of encrypted indices ***sEIndex*** of single keywords and encrypted indices ***iEIndex*** of keyword intersections.

The ***sEIndex*** update is based on the addition and deletion of the orthogonal vector set. Therein, if a keyword $w_u$ is replaced, its sub-index $sY_{w_u} = \boldsymbol{v}_u \cdot G_u \cdot P_u$ is deleted. If a keyword $w_u'$ is added, its sub-index $sY_{w_u}' = \boldsymbol{v}_u' \cdot G_u' \cdot P_u'$ is added. The index update algorithm for single keywords is expressed as follows:

$$\boldsymbol{sEIndex'} = \boldsymbol{sEIndex} - sY_{w_u} + sY_{w_u}' \tag{19}$$

The Data Owner then updates the encrypted indices of keyword intersections. If a keyword $w_u$ is replaced, ***iEIndex*** is traversed and elements corresponding to $w_u$ are deleted in the dictionary ${}_{iID}D\left(w_i\right)$ at first. Then, elements corresponding to keywords $w_1, w_2, \ldots, w_{u-1}$ in ${}_{iID}D\left(w_i\right)$ are traversed and their sub-indices $inY_{w_i} = \boldsymbol{v}_i \cdot G_{i\cap u} \cdot P_{i\cap u}$ are deleted. Finally, if a keyword $w_u'$ is added, it is inserted to the head of $\mathcal{W}$. In this way, $w_u'$ only need be subjected to an intersection operation with all other keywords in $\mathcal{W}$ to generate $inY_{w_u}'$, which is placed in the dictionary ${}_{iID}D\left(w_i\right)$. As a result, the new encrypted index $\boldsymbol{iEIndex}'$ of keyword intersections is generated. After integrating $\boldsymbol{iEIndex}'$ with $\boldsymbol{sEIndex}'$, $\boldsymbol{EIndex} = \left(\boldsymbol{sEIndex}', \boldsymbol{iEIndex}'\right)$ is updated.

## VI. SECURITY AND PERFORMACE ANALYSIS

The security and performance of the RBDC scheme are analyzed herein. The scheme is compared then with existing schemes from two perspectives: performance and function. Finally, the search efficiency and the index storage efficiency are assessed.

### A. SECURITY ANALYSIS

#### 1) SECURITY IN THE KNOWN CIPHERTEXT MODEL

In the known ciphertext model, attackers can establish a linear equation through the known ciphertexts, thus calculating the true values of encrypted indices and search trapdoors. Considering encrypted indices, the two parts ***sEIndex*** and ***iEIndex*** of the encrypted indices ***EIndex*** are both known to a Cloud Server, while the values of sub-indices corresponding to each keyword therein are unknown. ***sEIndex*** and ***iEIndex*** are separately encrypted using the random vector random vector $\boldsymbol{v}_i \in \mathcal{V}$, random number $r \in \mathcal{R}$, and GM and Paillier encryption algorithms, thus constructing a linear equation set:

$$\begin{cases} \sum_{i=1}^{|\mathcal{W}|} \left(\boldsymbol{v}_i \cdot Enc_{\mathrm{GM}}\left(ID_i\right) \cdot Enc_{\mathrm{PL}}\left(ID_i\right)\right) + \boldsymbol{v_r} \cdot r = \boldsymbol{sE\tilde{I}ndex}, \\ \sum_{j=i+1}^{|\mathcal{W}|} \left(\boldsymbol{v}_i \cdot Enc_{\mathrm{GM}}\left(ID_{i \cap j}\right) \cdot Enc_{\mathrm{PL}}\left(ID_{i \cap j}\right)\right) \\ \qquad\qquad + \boldsymbol{v_r} \cdot r = \boldsymbol{iE\tilde{I}ndex}, \end{cases} \tag{20}$$

where, considering that $\boldsymbol{v}_i, \boldsymbol{v}_r, r$ are all random, there are $(|\mathcal{W}\|\mathcal{D}|)$ and $|\mathcal{D}|$ unknown numbers separately comprising the left and right-hand sides of the equation. According to Eq. (20), the equation set contains $|\mathcal{W}|$ equations. In accordance with the properties of determinants, when there are more unknown numbers than determinants, the equation set does not have solutions, so data encrypted through GM and Paillier fail to be obtained through the equation set. That is to say, the true values of encrypted indices cannot be obtained. Likewise, true values of keyword data and encrypted data also cannot be attained through the search trapdoors. Therefore, the RBDC scheme ensures the privacy of data using the encryption mechanisms for indices and trapdoors.

The scheme also supports dynamic update of indices, so must also consider whether attackers can obtain relevant information about updated indices ***EIndex***$'$ from indices ***EIndex*** before updating (i.e. the forward security of the scheme). Indices before and after update are both known to a Cloud Server. At first, indices ***sEIndex***$'$ and ***sEIndex′*** are combined. Considering that $\boldsymbol{v}_i, \boldsymbol{v}_r, r$ are all random, while keywords $w_x$ and $w'_u$ before and after update are both encrypted using GM and Paillier encryption algorithms, which both contain random numbers, a $(2|\mathcal{W}|)$ matrix can be attained, which is composed completely of random numbers; because $|\mathcal{W}|$ is much greater than 2, the matrix is regarded as non-singular, so vector indices ***sEIndex*** and ***sEIndex′*** are proven to be linearly irrelevant. Similarly, it is also proven that two matrices, namely, indices ***iEIndex*** and ***iEIndex***$'$ of keyword intersections before and after update are linearly irrelevant. Hence, a Cloud Server cannot acquire relevant information about updated indices from those indices before updating, which ensures forward security of the scheme.

#### 2) SECURITY IN THE KNOWN BACKGROUND MODEL

Proofs in previous research [17] show that in the known background model, a Cloud Server can explore the relationship between encrypted indices and search trapdoors to mine and leak data thus breaching privacy rules for documents by analyzing the TF diagrams, thus inferring keyword information. When establishing indices in the RBDC scheme, various encrypted sub-indices are added for each index ***sEIndex*** of single keywords, to render it into a single vector. In addition, random numbers and random vectors are introduced, which ensures that the encrypted indices of single keywords are irrelevant to the inverted indices corresponding to keywords. Likewise, for each vector in the indices ***iEIndex*** of keyword intersections, two encryption methods are adopted for each index and the encrypted data are multiplied. Similarly, random numbers and random vectors are also introduced, so the relevance of multiple indices of keyword intersections cannot be obtained. In the same way, attackers cannot obtain search results by studying the relationship between search trapdoors. Meanwhile, random numbers are introduced to both Paillier and GM encryption methods. This means that even if the same search is repeated many times, a Cloud Server receives different indices and trapdoors, thereby countering any attack during statistical analysis and avoiding leakage of search modes. Therefore, the proposed scheme is secure in the known background model.

### B. PERFORMANCE ANALYSIS AND SCHEME COMPARISON

The section compares the RBDC scheme with existing searchable encryption schemes and explores two aspects of schemes (functions and performance).

Comparison of RBDC with other schemes is facilitated by referring to the data in TABLE 1: $M$ represents the length of the longest inverted indices generated by MRSE [4] and OXT [5]. $\#DB\,(w)$ denotes the length of inverted indices generated by schemes MRSE, OXT, and IBE [6]; *strg* refers to the storage space of indices. From the perspective of functions of schemes, RBDC supports multi-keyword Boolean search compared with IBE. Compared with MRSE and OXT, RBDC can rank searched files. RBDC supports dynamic update of keyword indices in comparison with MRSE, OXT, and IBE.

Regarding the performance of schemes, the time complexity of schemes is assessed at first. The keyword set to be searched is assumed to be $\mathcal{W} = \left(w_1, w_2, \ldots, w_q\right)$. At first, the single-keyword search trapdoor of each keyword is taken to have multiplication and power operations with the single-keyword indices, and the search efficiency is $O\left(|\mathcal{W}|\right)$. Then, the search trapdoor $it_{w_q}$ for intersections of keywords $w_1, w_2, \ldots, w_{q-1}$ is taken to have multiplication and power operations with the index ***iEIndex*** of keyword intersections and the search efficiency is $O\left(|\mathcal{W}|^2\right)$. Then, the time efficiency of the search algorithm in RBDC is $O\left(|\mathcal{W}|^2\right)$. Compared with MRSE and OXT that also support Boolean searching, RBDC improves the efficiency of the search algorithm. The time complexity of the search algorithm of RBDC is lower than that of IBE mainly because IBE only supports single-keyword search functions intersections. RBDC further improves the index storage efficiency; because

**TABLE 1.** Function and performance comparison between RBDC and other schemes.

| Scheme | Boolean | Ranking | Dynamic update | Search efficiency | Storage efficiency |
|---|---|---|---|---|---|
| MRSE | × | √ | √ | $O\left(\lvert W'\rvert^2 \lvert D\rvert\right)$ | $O\left(strg\left(\sum_w \#DB(w)\right)\right)$ |
| OXT | √ | × | × | $O\left(\lvert W'\rvert^2 M\right)$ | $O\left(strg\left(\sum_w \#DB(w)\right)+\sum_w strg\left(\sum_{v\in co(w)} \#DB(w)\cap DB(v)\right)\right)$ |
| IBE | √ | × | × | $O\left(\lvert W'\rvert^2 M\right)$ | $O\left(strg\left(\sum_w \#DB(w)\right)+\sum_w strg\left(\sum_{\substack{v\in co(w)\\ T_{w,v}>p}} \#DB(w)\cap DB(v)\right)\right)$ |
| RBDC | √ | √ | √ | $O\left(\lvert W'\rvert^2\right)$ | $O\left(strg(\#\boldsymbol{SEIndex})+strg\left(\lvert W\text{-}1\rvert \#\boldsymbol{iEIndex}\right)\right)$ |

the single-keyword index ***sEIndex*** is a vector element, the storage efficiency of the scheme is $O(strg(\#\boldsymbol{sEIndex}))$. As the index ***iEIndex*** of keyword intersections is a dictionary containing $(|\mathcal{W}|-1)$ vector elements, its storage efficiency is $O(strg(|\mathcal{W}-1|\,\#\boldsymbol{iEIndex}))$. Therefore, the storage efficiency of single-keyword indices of RBDC is only related to the length of vectors but does not increase with the increase in the number of keywords. Meanwhile, the storage space is also narrowed when storing indices of keyword intersections. When $|\mathcal{W}|$ is large, this substantially improves the index storage efficiency to a level higher than those of MRSE and OXT. IBE does not support Boolean searching so is not compared with others in terms of the index storage efficiency.

### C. EFFICIENCY TESTS

Experiments are undertaken to test the index storage space, search efficiency, ranking efficiency and dynamic update efficiency. A C/S RBDC prototype system is designed and realized using Java in the Win10 operating system.

The index storage space is tested: experimental results are shown in Figure 7, in which the abscissa and ordinate separately represent the number of keywords and the memory size. For single-keyword indices, the storage space of the proposed RBDC scheme does not change to any significant extent with the increasing number of keywords. This is because summation of vectors is adopted to store indices in a vector. For multi-keyword indices, the storage efficiency of RBDC is also significantly improved compared with IBE and OXT that also support Boolean search.

The keyword search efficiency is then assessed. At first, the single-keyword search efficiency of RBDC is tested and compared with that of MRSE (Figure 8). In the figure, the abscissa and ordinate separately represent the number of file sets to be searched and the search time. Experiments clearly show that the time complexity of RBDC does not increase with the growing number of file sets during single-keyword search of RBDC. Instead, the time complexity remains constant, that is, $O(|\mathcal{W}|)$. As the number of file sets grows, the single-keyword search efficiency of RBDC is improved significantly.

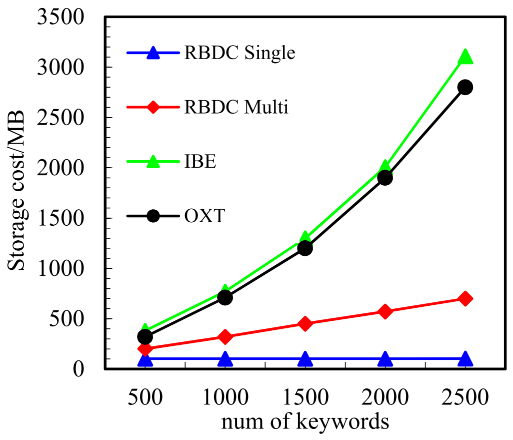

**FIGURE 7.** Index storage efficiency.

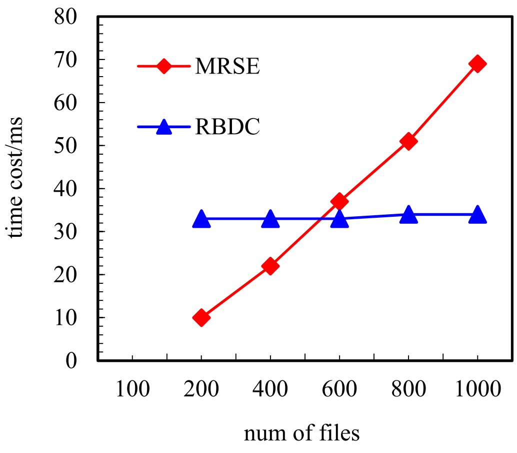

**FIGURE 8.** Single-keyword search efficiency.

Test results of multi-keyword Boolean search are illustrated in Figure 9. Assuming all Boolean searches entail an intersection operation, the abscissa and ordinate separately denote the number of keywords to be searched and the search time. Test results conclude that during multi-keyword Boolean searches, the time complexity of the search algorithm is $O\left(|\mathcal{W}|^2\right)$. Compared with OXT that also supports Boolean searching, the proposed scheme improves

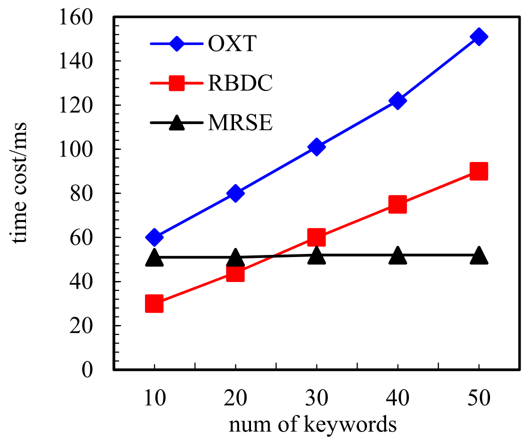

**FIGURE 9.** Multi-keyword Boolean search efficiency.

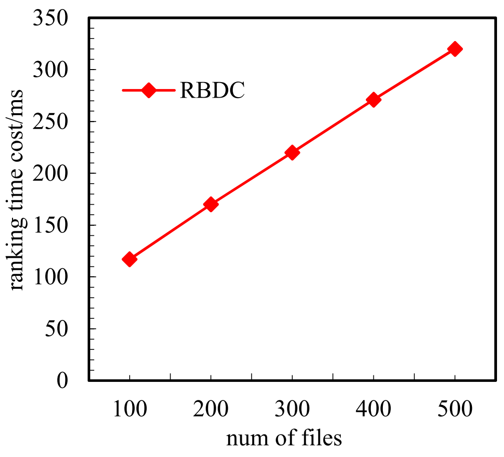

**FIGURE 10.** Ranking efficiency.

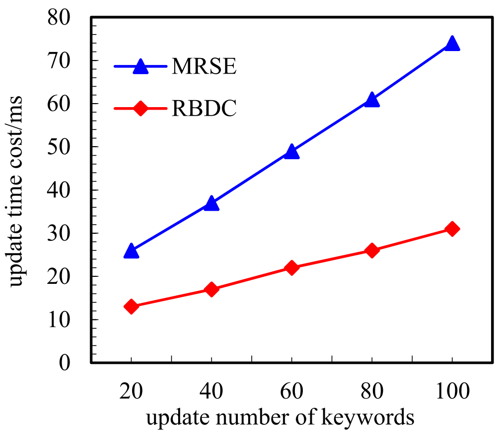

**FIGURE 11.** Dynamic update efficiency.

efficiency of the search algorithm; because MRSE does not support Boolean searching, its search efficiency is a constant that does not vary with the growing number of keywords in multi-keyword search. Considering this, the proposed scheme is superior to MRSE in multi-keyword search only when there are few keywords to be searched.

The ranking efficiency of schemes is then tested (Figure 10). Although MRSE supports rankable search, it is not compared with other schemes because ranking is involved in the search stage.

Finally, the update efficiency of schemes is assessed (Figure 11): because MRSE only supports ranking in single-keyword searches, only the updating of single-keyword indices of RBDC is tested and compared with that of MRSE. As for updating operations, only that updating of keywords is considered, while addition and deletion of keywords are ignored.

## VII. CONCLUSION

To solve the problem whereby most existing searchable encryption schemes do not support multi-keyword Boolean searching, the RBDC scheme is proposed. Based on traditional searchable encryption schemes, the scheme generates encrypted secure indices with high search efficiency and high storage efficiency using GM and Paillier encryption algorithms. Encrypted indices of single keywords and keyword intersections are then constructed according to the tenets of set theory to achieve multi-keyword Boolean searches. TF-IDF is used to construct the forward score indices, and the searched files are ranked by virtue of the third-party entity SCP. Meanwhile, the method also can dynamically update multiple keywords and improve the efficiency thereof. Thereafter, security analysis shows that the algorithm can counter two different types of security threat. Finally, the superiority of the RBDC scheme has been verified through function and performance analysis and comparison with other searchable encryption schemes.

## AKNOWLEDGMENT

The authors would like to thank IEEE Access journal and their respective Universities for their support.

## REFERENCES

[1] O. Goldreich and R. Ostrovsky "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, 1996.
[2] H.-A. Park, J. W. Byun, and D. H. Lee, "Secure index search for groups," in *Proc. 2nd Int. Conf. Trust, Privacy Secur. Digital Bus.*, Piscataway, NJ, USA, 2005, pp. 128–140.
[3] S. Tahir, S. Ruj, A. Sajjad, and M. Rajarajan, "Fuzzy keywords enabled ranked searchable encryption scheme for a public cloud environment," *Comput. Commun.*, vol. 133, pp. 102–114, Jan. 2019.
[4] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[5] Y. Zhang, Y. Zhao, Y. Wang, and Y. Li, "Searchable public key encryption supporting simple Boolean keywords search," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. E103A, no. 1, pp. 114–124, 2020.
[6] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Cham, Switzerland: Springer 2017, pp. 94–124.
[7] M. R. Senouci, I. Benkhaddra, A. Senouci, and F. Li, "An efficient and secure certificateless searchable encryption scheme against keyword guessing attacks," *J. Syst. Archit.*, vol. 119, Oct. 2021, Art. no. 102271.
[8] Z. Jiang, K. Zhang, L. Wang, and J. Ning, "Forward secure public-key authenticated encryption with conjunctive keyword search," *Comput. J.*, Jun. 2022.
[9] Q. Jiang, "Pbsx: A practical private Boolean search using Intel SGX," *Inf. Sci.*, vol. 521, pp. 174–194, Jun. 2020.
[10] M. Panda and A. Nag, "Plain text encryption using AES, DES and SALSA20 by Java based bouncy castle API on windows and Linux," in *Proc. 2nd Int. Conf. Adv. Comput. Commun. Eng.*, Piscataway, NJ, USA, May 2015, pp. 541–548.

[11] M. Zeng, K. Zhang, H. Qian, X. Chen, J. Chen, and Y. Mu, "A searchable asymmetric encryption scheme with support for Boolean queries for cloud applications," *Comput. J.*, vol. 62, no. 4, pp. 563–578, Apr. 2019.

[12] Z. Chen, F. Zhang, P. Zhang, and H. Zhao, "Multi-user Boolean searchable encryption supporting fast ranking in mobile clouds," *Comput. Commun.*, vol. 164, pp. 100–113, Dec. 2020.

[13] J. E. Nicaretta, D. M. B. Zapa, L. F. M. Couto, L. M. Heller, A. S. D. A. Cavalcante, L. B. Cruvinel, L. L. Ferreira, R. M. D. Nascimento, V. E. Soares, L. M. F. Borges, C. M. D. O. Monteiro, and W. D. Z. Lopes, "Rhipicephalus microplus seasonal dynamic in a Cerrado biome, Brazil: An update data considering the global warming," *Veterinary Parasitology*, vol. 296, Aug. 2021, Art. no. 109506.

[14] Y. Chen, "Searchable encryption system for big data storage," in *Proc. Int. Conf. Pioneering Comput. Scientists, Eng. Educators*, 2021, pp. 139–150.

[15] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, vol. 18, no. 1, pp. 186–208, Feb. 1989.

[16] X. D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, Piscataway, NJ, USA, May 2000, pp. 44–55.

[17] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Applied Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2005, pp. 442–455.

[18] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–15.

[19] L. Yang, Q. J. Zheng, and X. X. Fan, "RSPP: A reliable, searchable and privacy-preserving e-healthcare system for cloud-assisted body area networks," in *Proc. IEEE Conf. Comput. Commun.*, Piscataway, NJ, USA, May 2017, pp. 1–9.

[20] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert, "Bonsai trees, or how to delegate a lattice basis," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2010, pp. 523–552.

[21] H. Li, H. Tian, F. Zhang, and J. He, "Blockchain-based searchable symmetric encryption scheme," *Comput. Electr. Eng.*, vol. 73, pp. 32–45, Jan. 2019.

[22] M. Kim and K. Lauter, "Private genome analysis through homomorphic encryption," *BMC Med. Informat. Decis. Making*, vol. 15, no. S5, p. 3, Dec. 2015.

[23] R. Fagin, "Combining fuzzy information from multiple systems," *J. Comput. Syst. Sci.*, vol. 58, no. 1, pp. 777–780, 1996.

[24] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 614–656, Jun. 2003.

[25] N. Bruno, L. Gravano, and A. Marian, "Evaluating top-K queries over web-accessible databases," *ACM Trans. Database Syst.*, vol. 29, no. 2, pp. 319–362, 2004.

[26] C. C. Chang and S. W. Hwang, "Minimal probing: Supporting expensive predicates for top-K queries," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2002, pp. 346–357.

[27] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-K query processing techniques in relational database systems," *ACM Comput. Surveys*, vol. 40, no. 4, pp. 1–58, Oct. 2008.

[28] T. Mei, Y. Rui, S. Li, and Q. Tian, "Multimedia search reranking: A literature survey," *ACM Comput. Surveys*, vol. 46, no. 3, pp. 1–38, Jan. 2014.

[29] N. Bruno, S. Chaudhuri, and L. Gravano, "Top-k selection queries over relational databases," *ACM Trans. Database Syst.*, vol. 37, no. 2, pp. 33–67, 2002.

[30] A. Ghose, P. G. Ipeirotis, and B. Li, "Examining the impact of ranking on consumer behavior and search engine revenue," *Manag. Sci.*, vol. 60, no. 7, pp. 85–94, 2014.

[31] A. Vlachou, C. Doulkeridis, Y. Kotidis, and K. Norvag, "Monochromatic and bichromatic reverse top-k queries," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 8, pp. 1215–1229, Aug. 2011.

[32] Y. Miao, "Hybrid keyword-field search with efficient key management for industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3206–3217, Jun. 2019.

[33] Y. Miao, W. Zheng, X. Jia, X. Liu, K. R. Choo, and R. H. Deng, "Ranked keyword search over encrypted cloud data through machine learning method," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 525–536, Jan. 2023, doi: 10.1109/TSC.2021.3140098.

[34] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Adv. Cryptol. (EUROCRYPT)*. Berlin, Germany: Springer, 1999, pp. 223–238.

[35] K. Schmidt-Samoa and T. Takagi, "Paillier's cryptosystem modulo $p^2q$ and its applications to trapdoor commitment schemes," in *Proc. Int. Conf. Cryptol. Malaysia*, in Lecture Notes in Computer Science, vol. 14, no. 7, 2001, pp. 699–703.

[36] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.

[37] A. Degada and H. Thapliyal, "2-phase adiabatic logic for low-energy and CPA-resistant implantable medical devices," *IEEE Trans. Consum. Electron.*, vol. 68, no. 1, pp. 47–56, Feb. 2022.

**SIXU GUO** was born in 1996. He received the bachelor's degree from the School of Information Security, Northeastern University, and the master's degree from the School of Software Engineering, Northeastern University. He is currently pursuing the Ph.D. degree with Dalian Jiaotong University. He is also an Information Security Engineer with the Security Institute, China Mobile Research Institute. His research interests include cryptography, privacy computing, and searchable encryption.

**HUIZHENG GENG** was born in 1988. He received the bachelor's and master's degrees from the School of Mathematics, Shandong University. He is currently the Technical Manager of the Security Institute, China Mobile Research Institute. His research interests include the research of mobile communication network security and data security of plasma sources, and the fabrication of micro- or nanostructured surfaces.

**LI SU** was born in 1981. He received the Ph.D. degree. He is currently the Vice Director of the Security Technology Department, China Mobile Research Institute. He is also a Professor-Level Senior Engineer. He has 14 years of experience in the field of information security of telecommunications. He holds the certificate of Certified Information Systems Security Professional (CISSP). His research interests include network security and information security.

**SHEN HE** was born in 1980. He received the Ph.D. degree. He is currently a Senior Engineer. His research interests include network security, data security, mobile internet security, trusted computing, and other aspects of research.

**XINYUE ZHANG** was born in 1994. She received the bachelor's and master's degrees in information security from Northeast University, with a focus on searchable encryption and privacy computing.