Hindawi
Security and Communication Networks
Volume 2020, Article ID 8893016, 13 pages
https://doi.org/10.1155/2020/8893016

Research Article

A Dynamic Searchable Symmetric Encryption Scheme for Multiuser with Forward and Backward Security

Xi Zhang,[1] Ye Su,[1] and Jing Qin[1,2]

[1]*School of Mathematics, Shandong University, Jinan 250100, China*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*

Correspondence should be addressed to Jing Qin; qinjing@sdu.edu.cn

Received 30 June 2020; Revised 23 August 2020; Accepted 24 September 2020; Published 20 October 2020

Academic Editor: A. Peinado

Dynamic Searchable Symmetric Encryption for Multiuser (M-DSSE) is an advanced form of symmetric encryption. It extends the traditional symmetric encryption to support the operations of adding and deleting the encrypted data and allow an authenticated group of data users to retrieve their respective desired encrypted data in the dynamic database. However, M-DSSE would suffer from the privacy concerns regarding forward and backward security. The former allows an attacker to identify the keywords contained in the added data by lunching file-injection attacks, while the latter allows to utilize the search results and the deleted data to learn the content. To our knowledge, these privacy concerns for M-DSSE have not been fully considered in the existing literatures. Taking account of this fact, we focus on the dynamic searchable symmetric encryption for multiuser meeting the needs of forward and backward security. In order to propose a concrete scheme, the primitives of Pseudorandom Functions (PRF) and the Homomorphic Message Authenticator (HMAC) are employed to construct the inverted index and update the search token. The proposed scheme is proven secure in the random model. And the performance analysis shows that the proposed scheme achieves the enhanced security guarantees at the reasonable price of efficiency.

1. Introduction

Searchable encryption (SE) is popular among the various cloud storage services because one can keep the ability to selectively retrieve the encrypted data that he or she stored on the cloud. And there are many traditional works [1–5] focusing on it. However, the traditional works cannot meet the using needs or habits of the clients. It is because frequently clients upload some new data and delete or modify some encrypted data in cloud and many clients often share data with others. For example, a regional medical center needs to update the local Electronic Health Records (EHRs) [6] periodically and share them with other medical institutions to conduct advanced research. This means that the regional medical center should have the ability of updating the encrypted data and authorizing others to search over his data storage in cloud.

To address it, some dynamic searchable encryption schemes [7–11] have been proposed to support data updates. These works are designed for the single client that means only the client itself can search on the cloud and cannot meet the requirement to share data. Data sharing is widely used by both individuals and organizations, and we introduce two forms here: one is multiwriter/multireader and the other is single writer/multireader. For ease of exposition, we call the client who owns data the data owner and the clients who share the data the data user. The former means many data owners and many data users, and there have been efforts to design schemes for fine-grained keyword search [12, 13], privacy-preserving attribute-based keyword search [14], and rank keyword search in arbitrary language [15]. The latter means only one data owner and many data users who are especially popular among companies, schools, and medical institutions. It was named as dynamic searchable symmetric encryption for multiuser (M-DSSE), which was also called multiclient in [16], and it is the topic that we are interested in.

However, M-DSSE suffers from the privacy concerns regarding forward and backward security. The former is that the adversary can use the file-injection attacks to compromise the privacy of the data and keywords. This attack was proposed by Zhang et al. [17] in 2016. More specifically, one can inject the carefully selected files and trick the client into encrypting them, then identify the keyword by matching the submitted search tokens and injected files, and consequently get all files containing this keyword. This behavior seriously undermines the privacy of data, for example, the disclosure of the patient's identity information or home address in EHRs that should be kept private. The latter means that, in most searchable encryption schemes, the identifier of the deleted document still can be retrieved by the server in the subsequent search. Then when the regional medical center deletes some EHRs, not only the privacy of the patients will be disclosed, but also the accuracy of the advanced research will be affected.

Both attacks are simple but destructive. Therefore, M-DSSE needs stronger security from a practical and safe point of view, that is, Dynamic Searchable Symmetric Encryption for Multiuser with Forward and Backward Security (FBM-DSSE). Although there are many works concentrating on the dynamic symmetric searchable encryption schemes with forward and backward security (FB-DSSE) [18–22], most of them cannot be extended to FBM-DSSE directly. On the one hand, if the method used in FB-DSSE is extended to FBM-DSSE directly, the data owner may need to share the whole key with all users. And users can do more rather than search only. Specifically, users can change client's data without restrictions and destroy data's integrity and privacy. On the other hand, the existing FB-DSSE schemes adopt some special structures to achieve forward and backward security, which is not suitable to extend to multiuser setting. For example, Li et al. [23] reduce information leakage in SSE by partitioning the inverted index into disjoint partitions and generating subkeyword sets. When searching, the data owner chooses subkeyword according to his own needs and the server needs to delete all the touched blocks after each search. This search method is complex for users in M-DSSE and may get incomplete search results due to untimely updates. Therefore, it is not feasible to directly implement the methods of FB-DSSE in FBM-DSSE, and it is still a problem that how to achieve forward and backward security in M-DSSE.

Considering the serious consequences that these security issues may bring and the inflexibility of the methods from FB-DSSE to FBM-DSSE, we believe that solving security problems in FBM-DSSE is of great practical significance. Taking account of it, we focus on the Dynamic Searchable Symmetric encryption schemes supporting multiuser with Forward and Backward Security. In order to achieve this it, we give a concrete FBM-DSSE scheme based on the Pseudorandom Functions (PRF), the Homomorphic Message Authenticator (HMAC) [24], and the bitmap index [25, 26].

Our contributions are summarized as follows:

The proposed scheme is forward and backward secure. We combine the homomorphic MAC and the bitmap index to achieve secure and efficient updates of the search token and use PRF to hide the specific correspondence between files and indexes to protect the update information from being leaked. Specifically, we use pseudorandom functions to reorder files and the keywords in index and the server could not identify the specific relationship between them so that it could not get any private information except the current search results.

The proposed scheme has rich functionality. It is obvious that the proposed scheme supports update operations and multiuser setting. It also supports verifiability and can realize revocation of the user's access permission. Specifically, if the data owner wants to cancel someone's access right to the updated file but retain the permission for the previously searched file, he or she is not needed to send the updated search token to the user. Furthermore, our scheme is easily extended to support multifunctional search such as wildcard search [27], similarity search (including hamming distance and edit distance), fuzzy keyword search [28], and disjunctive [29] (or conjunctive [30]) keyword search.

The proposed scheme has a comprehensive security analysis. We give the correctness of the scheme and the rigorous security proof of forward and backward security according to the definitions in DSSE. And the security of the proposed scheme can be reduced to the existence of pseudorandom functions and the CPA-secure symmetric encryption system, which provides a concrete implementation favorable guarantee.

The paper is organized as follows. Section 1 is the introduction. Section 2 introduces the related work. Cryptographic tools and notations are introduced in Section 3. Section 4 presents the system model, security model, and the definition of the forward security and the backward security. Section 5 mainly introduces the proposed scheme and the security analysis. Section 6 shows simply how the proposed scheme can be extended to support multifunctional search. Section 7 gives the experiment result and its analyses. At last, the paper is concluded in Section 8.

2. Related Works

Searchable encryption (SE) is popular among various cloud storage services because it protects plaintext information from being leaked to the compromised server while preserving the search functionality. There are two areas in SE: public key encryption with keyword search (PEKS) [2–5] and searchable symmetric encryption (SSE). In our work, we mainly talk about the SSE.

The first symmetric searchable encryption (SSE) was proposed by Song et al. [1] in 2000. They proposed a special two-layer encryption scheme. Because this scheme needs to scan the file, the searching time is linear to the length of the files. The followed work is that of Curtmola et al. [31]. They constructed the first inverted index and achieved that the amount of the server's work is proportional to the number of

files containing the queried keyword. There are also many SSE schemes concentrating on rich queries [32–34]. However, those works mainly focused on searchable encryption under static conditions, that is, there is no update of files. Due to the universality of dynamic operation, it is important and necessary to migrate it to the cloud services. Therefore, the dynamic searchable encryption (DSSE) is more in line with the practical situation.

In 2012, Kamara et al. [7] constructed a DSSE scheme based on the inverted index technique and achieved sublinear search complexity and CKA2-secure. And then they [8] constructed another dynamic searchable encryption scheme based on the red-black tree index and achieved parallel search of keywords and parallel addition and deletion of files. There are others scheme including Naveed et al. [9] based on the blind storage, Xia et al. [10] based on the tree-based index, and Guo et al. [11] based on the inverted index. At the same time, DSSE schemes leak some information such as search pattern (the pattern in search queries), size pattern [18] (the number of search results), and access pattern (how the encrypted data or indexes are accessed). These attracted people's attention. Zhang et al. [17] proposed a file-injection attack in 2016, and the attacker can determine the keywords corresponding to the token by injecting files containing different keywords. The effective attack calls for the stronger security of DSSE. The trivial way of downloading and then decrypting the full encrypted files to obtain the needed files contradicts the purpose of search encryption. The ways of using secure two-party computation, fully homomorphic encryption, and oblivious RAM to realize the higher level of security waste more local storage space and acquire high computational and communication complexity. Both of them are very expensive and impractical.

In 2014, the term of forward privacy and backward privacy was first proposed by Stefanov et al. [18], and it is the new secure goal that dynamic searchable encryption schemes should meet in the practical level. Since 2014, some schemes have been proposed to achieve it using different methods, including but not limited to Stefanov et al.[18] based on a hierarchical structure of logarithmic levels; Bost [19] based on trapdoor permutations, and then they proposed a forward and backward scheme relying on primitives such as constrained pseudorandom functions and puncturable encryption schemes[35]; Wang et al.[20] based on the proxy server; Sun et al.[21] based on the symmetric puncture encryption primitive; and Kim et al.[22] based on dual dictionary.

As for the DSSE for multiusers, some works have been carried out. As for multiwriter/multireader, Nair and Rajasree [13] used a bilinear accumulator to implement a fine-grained multiuser solution for search control and access control; Popa and Zeldovich [36] proposed a method for encrypting different files with different keys. As for single writer/multireader, Curtmola et al. [31] proposed the first scheme structure based on broadcast encryption; Wang et al. [20] proposed a multiuser forward secure dynamic searchable symmetric encryption in 2018; Jarecki et al. [37] used the forgotten PRF to generate keyword trapdoors. However, the research on the Dynamic Searchable Symmetric Encryption for multiusers with Forward and Backward Security (FBM-DSSE) is not enough.

3. Cryptographic Tools and Notions

3.1. Cryptographic Tools

3.1.1. Pseudorandom Functions. There are security parameter λ and κ polynomial in λ. There are key space $\mathscr{K} = \{0,1\}^{\kappa}$, domain $\mathscr{D}$, and output space $\mathscr{R}$. Let F: $\mathscr{K} \times \mathscr{D} \longrightarrow \mathscr{R}$ be a keyed function. We define

$$\mathrm{Adv}^{\mathrm{PRF}}_{\mathscr{A},F}(\lambda) = \left|\Pr\left[\mathscr{A}^{F_K(\cdot)}\left(1^{\lambda}\right) = 1 \colon K \longleftarrow \mathscr{K}\right] - \Pr\left[\mathscr{A}^{\phi(\cdot)}(y) = 1\right]\right|, \tag{1}$$

as the advantage of the adversary $\mathscr{A}$ against the pseudorandomness of F, where K is a random element in key space and ϕ: $\mathscr{D} \longrightarrow \mathscr{R}$ is chosen in all functions from $\mathscr{D}$ to $\mathscr{R}$. If for any PPT adversary $\mathscr{A}$, the advantage $\mathrm{Adv}^{\mathrm{PRF}}_{\mathscr{A},F}(\lambda)$ is a negligible function; then, we say that F is a negligible function.

3.1.2. Homomorphic MAC. In this paper, we will use a construction of a homomorphic message authenticator scheme (HMAC) allowing for homomorphic evaluation and arbitrary composition (i.e., outputs of previously authenticated computations can be used as inputs for new ones). The scheme is simple and efficient and its security relies only on a pseudorandom function.

A HMAC is 4-tuple of algorithms working as follows:

Setup: $1^{\lambda} \longrightarrow (\mathrm{sk}, \mathrm{ek})$. For a security parameter 1^{λ}, the algorithm outputs the secret key *sk* and evaluation key *ek* needed in the scheme.

Auth: $(\mathrm{sk}, \tau, m) \longrightarrow \sigma$. This algorithm inputs the secret key sk, a label τ, and a message $m \in M$, and it outputs the corresponding tag σ.

Ver: $(\mathrm{sk}, m, P, \sigma) \longrightarrow 0$ or 1. This algorithm inputs the secret key sk, a program $P = (f, \tau_1, \ldots, \tau_n)$, a message m, and its tag σ, and it outputs 0 (reject) or 1 (accept).

Eval: $(\mathrm{ek}, f, \sigma_1, \ldots, \sigma_n) \longrightarrow \sigma$. This algorithm inputs the evaluation key sk, a circuit f: $M^n \longrightarrow M$, and $\sigma_1, \ldots, \sigma_n$, and it outputs a new tag σ.

We restrict that the arithmetic circuits f used in the proposed scheme only has the additive gates, so the size of the produced tags will not grow. The concrete description of the HMAC scheme is shown in [24].

3.1.3. Symmetric Encryption. A symmetric encryption consists of the following algorithms:

Gen: $1^{\lambda} \longrightarrow \mathrm{sk}$. For a security parameter 1^{λ}, the algorithm outputs the secret key sk needed in the scheme.

Enc: $(\mathrm{sk}, m) \longrightarrow c$. This algorithm inputs the secret key sk and a message $m \in M$, and it outputs the

corresponding ciphertext c. Since Enc may be randomized, we write this as $c \longleftarrow \mathrm{Enc}_{sk}(m)$.

Dec: (sk, c). This algorithm inputs the secret key sk and ciphertext c, and it outputs m or $\perp$. We assume that Dec is deterministic, and so write $m := \mathrm{Dec}_{sk}(c)$.

CPA-Secure: there is a symmetric encryption $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, and the IND-CPA experiment is shown below.

$\mathrm{Privk}^{\mathrm{CPA}}_{\mathscr{A},\Pi}(n)$:

(1) $k \longleftarrow \mathrm{Gen}(1^n)$.

(2) Send 1^n to adversary $\mathscr{A}$, and $\mathscr{A}$ can ask the random oracle $c \longleftarrow \mathrm{Enc}_{sk}(m)$. At last, $\mathscr{A}$ outputs m_0, m_1, where $|m_0| = |m_1|$.

(3) $b \longleftarrow_R \{0,1\}$, $c \longleftarrow \mathrm{Enc}_{sk}(m_b)$ and send it to $\mathscr{A}$, where c is named to the challenge ciphertext.

(4) $\mathscr{A}$ continues to ask the random oracle $c \longleftarrow \mathrm{Enc}_{sk}(m)$ and outputs a bit b'.

(5) If $b' = b$, then $\mathrm{Privk}^{\mathrm{CPA}}_{\mathscr{A},\Pi}(n) = 1$, otherwise $\mathrm{Privk}^{\mathrm{CPA}}_{\mathscr{A},\Pi}(n) = 0$.

Definition 1. A symmetric encryption $\Pi = \{\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec}\}$ is CPA-Secure only if for all Probability Polynomial adversary $\mathscr{A}$, and there exists a negligible function negl:

$$\Pr\left[\mathrm{Privk}^{\mathrm{CPA}}_{\mathscr{A},\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathrm{negl}(n). \tag{2}$$

3.2. Notations. We show some notations used in the proposed scheme in Table 1.

4. System Model and Security Model

4.1. System Model and Security Model. As shown in Figure 1, the system model consists of three entities: the data owner, data users, and the cloud server.

4.1.1. Data Owner. The data owner extracts the keywords from files and constructs the plain index. Then, he encrypts all files and the index with different private keys and uploads the encrypted EDB to the cloud server. Besides that, he is also responsible for managing users and updating files. More specifically, he distributes keys, updates tokens to users, and sends the updated encrypted files and index to the cloud server. And the data owner is assumed to be always trusted.

4.1.2. Cloud Server. The main job of the cloud server is to store the encrypted files and index from data owner and perform searches for data users. When receiving the updated information from the data owner, it updates the encrypted database. Upon receiving the search requests from data users, it performs search operation over the index and returns the search results to data users. The cloud server is regarded as an honest but curious entity. That is to say, it performs algorithms honestly but will try its best to get more valuable information. Besides, the cloud server may return invalid or nonupdated search results to the data user because of computation mistakes.

4.1.3. Data Users. Data users are authorized and shared with some keys needed in the scheme by the data owner. When they want to search files containing the interested keyword, they send the search token to cloud server and receive the search results. Furthermore, they can verify the validity of the results with the help of the data owner. The data users are assumed to be always trusted.

4.2. Security Model. We use two games $\mathrm{DSSEREAL}^{\Gamma}_{\mathscr{A}}(1^\lambda)$ and $\mathrm{DSSEIDEAL}^{\Gamma}_{\mathscr{A},\mathscr{S}}(1^\lambda)$ to show the security definition of DSSE. The $\mathrm{DSSEREAL}^{\Gamma}_{\mathscr{A}}(1^\lambda)$ is the same as the DSSE. And the $\mathrm{DSSEIDEAL}^{\Gamma}_{\mathscr{A},\mathscr{S}}(1^\lambda)$ is conducted by simulator $\mathscr{S}$ with the leakage of DSSE. The leakage of DSSE is parameterized by a function $\mathscr{L} = (\mathscr{L}^{\mathrm{Stp}}, \mathscr{L}^{\mathrm{Srch}}, \mathscr{L}^{\mathrm{Updt}})$, which describes the information leaked to the adversary $\mathscr{A}$. The adversary $\mathscr{A}$ will interact with $\mathrm{DSSEREAL}^{\Gamma}_{\mathscr{A}}(1^\lambda)$ or $\mathrm{DSSEIDEAL}^{\Gamma}_{\mathscr{A},\mathscr{S}}(1^\lambda)$ and guess. If the adversary $\mathscr{A}$ can correctly guess the game he interacts only with a negligible advantage, then we can say the DSSE is secure because of the leaked information limited to the leakage function $\mathscr{L}$.

$\mathrm{DSSEREAL}^{\Gamma}_{\mathscr{A}}(1^\lambda)$: the adversary $\mathscr{A}$ chooses a database DB and inputs it, then this game performs the $\mathrm{Setup}(1^\lambda, \mathrm{DB})$ and outputs EDB. During the search phase, the adversary $\mathscr{A}$ runs search query q or update query $(\mathrm{op}, \mathrm{in})$, where op is the operation and the in is the identifier of the file. The game outputs the search results by performing $\mathrm{Search}(q)$ or $\mathrm{Update}(\mathrm{op}, \mathrm{in})$. Eventually, $\mathscr{A}$ outputs a bit.

$\mathrm{DSSEIDEAL}^{\Gamma}_{\mathscr{A},\mathscr{S}}(1^\lambda)$: the adversary $\mathscr{A}$ chooses a database DB and inputs it; then, the simulator $\mathscr{S}$ performs the $\mathscr{L}^{\mathrm{Stp}}(1^\lambda, \mathrm{DB})$ and outputs EDB. During the search phase, the adversary $\mathscr{A}$ runs search query q or update query $(\mathrm{op}, \mathrm{in})$. The simulator $\mathscr{S}$ outputs the search results by performing the leakage function $\mathscr{L}^{\mathrm{Srch}}(q)$ or $\mathscr{L}^{\mathrm{Updt}}(\mathrm{op}, \mathrm{in})$. Eventually, $\mathscr{A}$ outputs a bit.

Definition 2. A DSSE scheme Γ is $\mathscr{L}$*-adaptively secure* only if for every PPT adversary $\mathscr{A}$, and there exists an efficient simulator $\mathscr{S}$ and a negligible function negl such that

$$\left|P\left[\mathrm{DSSEREAL}^{\Gamma}_{\mathscr{A}}\left(1^\lambda\right) = 1\right] - \Pr\left[\mathrm{DSSEIDEAL}^{\Gamma}_{\mathscr{A},\mathscr{S}}\left(1^\lambda\right) = 1\right]\right| \leq \mathrm{negl}\left(1^\lambda\right). \tag{3}$$

4.3. Forward and Backward Security. In 2016, Bost [19] defined the traditional forward privacy that the server cannot link the newly updated files with previously searched keywords. And in Li et al.'s work [23], they further defined the forward update privacy, strong forward search privacy, and weak forward search privacy. The forward update privacy requires that the information leaked in update

Table 1: Notations (used in our scheme).

Notation	Description
P_1	A secure keyed PRF used to generate the key for HMAC.Auth algorithm
P_2	A secure keyed PRF used to generate the key for HMAC.Eval algorithm
$\{R\}$	A secure PRF family used to generate the file's column
l	Maximum number of files corresponding to security parameters
H	A secure keyed PRF used to generate the key of column
H_1	A secure PRF used to generate the keyword's hash
$\{Q\}$	A secure PRF family used to generate the row order of keywords
W	The set of keywords extracted from the file
F	The set of files
c	To column c of the index $1 \le c \le l$
k_c	The key corresponding to column c of the index
C_c	The ciphertext of the file corresponding to the cth column of the index
C_F	The ciphertext of the set of the files F
C_{F^t}	The ciphertext of the set of the files F at time t
α_i^t	Plaintext of row i of index at time t
τ	The input label of HMAC
F'	The file set that does not update
f_c^t	At time t, the plaintext of the file corresponding to column c of index
ΔF	$\Delta F = \{f, \mathrm{op}(f)\}$
ΔW	$\Delta W = \{w, \mathrm{op}(w)\}$

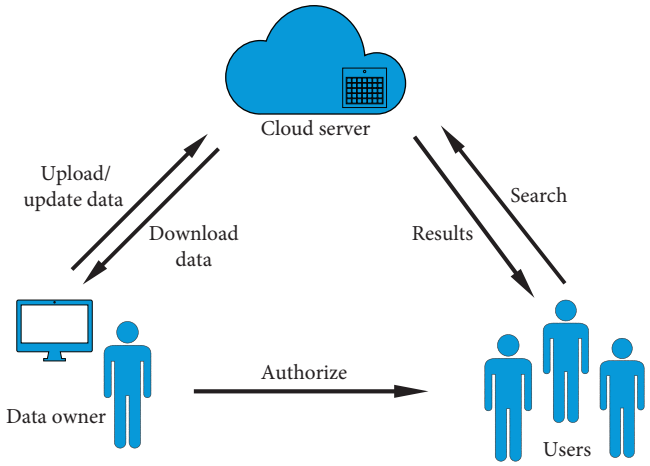

Figure 1: Architecture of our scheme.

operation should not be more than the identifier and the number of keywords of newly updated files. And the strong forward search privacy implies fully oblivious search operation, which is a too strong notion to achieve unless using the expensive protocols such as ORAM or PIR. The weak forward search privacy means the leaked information is the list of files containing the keyword w at the time t. Our scheme can achieve forward update privacy and weak forward search privacy, but for the consistent with most studies, we adopt the traditional forward privacy which is defined as follows.

Definition 3. A $\mathscr{L}$ *-adaptively secure* DSSE scheme Γ is forward secure if the update leakage function $\mathscr{L}^{\mathrm{Updt}}$ can be written as

$$\mathscr{L}^{\mathrm{Updt}}(\mathrm{op}, \mathrm{in}) = \mathscr{L}'(\mathrm{op}, (\mathrm{ind}_i, \mu_i)), \tag{4}$$

where (ind_i, μ_i) is the set of modified documents paired with number μ_i of modified keywords for the updated document ind_i. Specially, the leakage function $\mathscr{L}^{\mathrm{Updt}}(\mathrm{op}, w, \mathrm{bs}) = \mathscr{L}'(\mathrm{bs})$ in this paper.

In 2014, the term of backward privacy was first proposed by Stefanov et al. [18], and it was clearly defined by Bost et al. [35] in 2017. They defined three backward privacy from Type – I to Type – III, and Zuo et al. [26] formulated the most secure definition Type – I$^-$ in 2019. Our construction will adopt the latter, which is defined as follows.

Definition 4. A $\mathscr{L}$ *-adaptively secure* DSSE scheme Γ is Type – I$^-$ backward private only if the search and update leakage function $\mathscr{L}^{\mathrm{Srch}}$ and $\mathscr{L}^{\mathrm{Updt}}$ can be written as

$$\begin{aligned}\mathscr{L}^{\mathrm{Updt}}(\mathrm{op}, w, \mathrm{ind}) &= \mathscr{L}'(\mathrm{op}),\ \mathscr{L}^{\mathrm{Srch}}(w) \\ &= \mathscr{L}''(\mathrm{sp}(w), \mathrm{rp}(w), \mathrm{Time}(w)),\end{aligned} \tag{5}$$

where t is a timestamp, $\mathrm{sp}(w) = t : (t, w) \in Q$ is a search pattern, $\mathrm{rp}(w) = \mathrm{bs}$ represents all file identifiers that currently match w, and $\mathscr{L}'$ and $\mathscr{L}''$ are stateless.

5. The Proposed Scheme

5.1. Overview. We mainly consider how to achieve forward and backward security when the cloud server is semihonest and the users are honest (the collusion between the cloud server and the user is not considered here). In order to achieve such a security goal, we use a pseudorandom function to shuffle the order of files so that the cloud server cannot identify the specific relationship between the index

and the files from the search results and cannot get any private information.

First, it is necessary to ensure that the search token of the keywords will be changed after the update; otherwise, the cloud server will learn whether the newly updated documents match a previously searched keyword or not. Here, in order to reduce the computational complexity of the index update and facilitate the user to update the search token, we adopt the HMAC technology. The plain index is regarded as the message m and HMAC.Auth(m) is regarded as the symmetric key of the encrypted m. In this case, the data owner only needs to send HMAC.Auth(Δm) to the user and the user invokes the HMAC.Eval algorithm to get the latest key, which can be used to generate a search token.

Second, in order to not let the cloud server obtain the specific relationship between the index and the files, we assign a key to each column. When an update occurs, we use PRF to reorder the columns corresponding to the files and re-encrypt those files with the new key. It can be seen that our scheme is more suitable for scenarios where the ratio of file updated is relatively large.

Third, there will still be cases where the file has not been updated and the relationship with the column of the index has not changed. The adversary cannot distinguish them because the index has been updated with the CPA-secure symmetric encryption scheme.

5.1.1. The Bitmap Index. In this article, we use the bitmap index which is an inverted index with $O(1)$ search time. Specifically, we first extract keywords from the files. The bitmap index is a matrix of 0 and 1. The columns of the matrix correspond to different files. The rows of the matrix correspond to the keywords. If the number of the ith row and the jth column of the matrix is 1, it means that the keyword is included in the file, otherwise vice versa. The advantage of the bitmap is that it can easily implement the update of the index.

Take a simple example for simplicity, and there are 5 files at time t. The index of the keyword w corresponding to the ith row is represented by a binary string 01001, which indicates that the files corresponding to the 2nd and 5th column contain w. At time $t+1$, the index of w is updated and become 01100, which means that the files corresponding to columns 2 and 3 now contain w, and the file corresponding to column 5 does not contain w any more. In order to update the index and search token, the data owner only needs to change the index from 01100 to 01001, which is an easy operation.

5.1.2. The Homomorphic Message Authenticator. When data updates, the data owner needs to update the search token and send it to data users otherwise they cannot search on cloud any more. In order to explain clearly and simplicity, we still use the above example. The data owner updates the search token by calculating 01100 – 01001and sends the HMAC.Auth(01100 – 01001) to data users. Then, users execute HMAC.Eval and get the new search token. Since only the data users have the evaluation key, the interaction does not need the secret channel. Furthermore, the communication overhead is only a HMAC tag which efficiently reduces the transmission complexity and the whole process is simple.

5.1.3. The Pseudorandom Functions. The Pseudorandom Functions is mainly used to generate the bitmap index. For example, there are 5 files f_1, f_2, f_3, f_4, f_5 and 3 keywords w_1, w_2, w_3. And in time t, the files are mapped to $(3,4,5,1,2)$ and the keywords to $(3,1,2)$ by pseudorandom functions. The former means the first column of index indicates f_3, the second one is f_4, and so on. The latter shows the first row of index indicates w_3, the second one is w_1, and the last is w_2.

5.2. Concrete Construction. Now, we are ready to give our dynamic searchable symmetric encryption scheme for multiuser with forward and backward security. See Algorithm 1 for more information. Our scheme is based on the framework of DSSE = (Setup, Update, Search) and calls HMAC = (HMAC.Setup, HMAC.Auth, HMAC.Ver, HMAC.Eval) and keyed PRF. The scheme is defined by Algorithm 1.

Setup: $1^\lambda \longrightarrow (\mathrm{PK}, \mathrm{SK})$

The algorithm is run by the data owner. For a security parameter 1^λ, the algorithm outputs the PK and SKneeded in the scheme.

Update: $(\mathrm{SK}, \mathrm{EDB}^t, \Delta F, \Delta W) \longrightarrow (\mathrm{EDB}^{t+1}, \Delta\sigma_w^t)$

The algorithm is run by the data owner too. At the beginning, the initial index and EDB are empty, so when $t=0$, it means that it is the first time for the data owner to add the file. When $t \neq 0$, it means the normal update operations. It should be noted that, in order to be consistent with the DSSE structure, we have omitted the algorithm of adding users in Algorithm 1 FBM-DSSE, and the algorithm is defined as Algorithm 2: Adduser.

Search: $(t, UL, w, EDB^t) \longrightarrow C_w^t$

This polynomial time algorithm is executed by the cloud server and the users. When data users want to search w, he runs the line 1 to 8, generates the search token $Tr_u(w)$, and sends it to cloud server. Then, the cloud server verifies the legitimacy of the user and performs a search. Finally, the cloud server outputs the search results.

Actually, we have omitted the decryption and verification algorithms in Algorithm 1 FBM-DSSE for the same reason. And the algorithms are defined as Algorithm 3: Decrypt and Algorithm 4: Verify.

5.3. Security Analysis. In this section, we first present the correctness of the proposed scheme and then give the security analysis.

5.3.1. Correctness. if the user wants to search for files containing the keyword w at time t, he executes lines 1–8 of the search algorithm and then sends the search token of w to the cloud server. The ciphertext index saved in cloud is generated by the same pseudorandom function, which ensures the correctness of the searched keywords. Secondly, according to the bitmap index generation process and the

Setup: $1^{\lambda} \longrightarrow (\mathrm{PK}, \mathrm{SK})$
(1) input a security parameter λ
(2) generate $(P_1, P_2, \{R\}, H, H_1, \mathrm{WK}, \mathrm{WEK}, \mathrm{FK}, \{\mathrm{Q}\})$
(3) set $\mathrm{PK} = (P_1, P_2, H, H_1)$
(4) set $\mathrm{SK} = (\mathrm{WK}, \mathrm{WEK}, \mathrm{FK}, \{R\}, \{\mathrm{Q}\})$
Update: $(\mathrm{SK}, \mathrm{EDB}^t, \Delta F, \Delta W) \longrightarrow (\mathrm{EDB}^{t+1}, \Delta\sigma_w^t)$
data owner:
(1) if $t = 0$ then
(2) extract keywords $W = \{w\}_n$ from the files F^t
(3) attach keys with column identifiers c $(0 \le c \le l)$
(4) for each column identifier c
(5) $k_c \longleftarrow H(\mathrm{FK}, c)$
(6) (match files to column identifiers and encrypt files)
(7) for each file $f \in F^t$
(8) $c \longleftarrow R(f)$
(9) $C_c \longleftarrow \mathrm{Enc}_{k_c}(f)$
(10) if c that does not match files
(11) $r_c \longleftarrow_R \{0,1\}^*$
(12) $C_c \longleftarrow \mathrm{Enc}_{k_c}(r_c)$
(13) $C_{F^t} \longleftarrow C_{F^t} \bigcup C_c$
(14) (generate bitmap index)
(15) for each keyword $w \in W^t$
(16) $k_w \longleftarrow P_1(\mathrm{WK}, H_1(w))$
(17) $\mathrm{ek}_w \longleftarrow P_2(\mathrm{WEK}, H_1(w))$
(18) $\alpha_w^t = (a_{w1}, a_{w2}, \ldots, a_{wl})$:
(19) $a_{\mathrm{wc}} = 1 \Longleftrightarrow w \in f \; and \; R(f) = c$
(20) $\sigma_w^t \longleftarrow \mathrm{HMAC.Auth}(k_w, \tau, \alpha_w^t)$
(21) $\beta_w^t \longleftarrow \mathrm{Enc}_{\sigma_w^t}(\alpha_w^t)$
(22) $i \longleftarrow Q^t(\sigma_w^t)$
(23) $(A^t)^T = (\alpha_1^t, \alpha_2^t, \ldots, \alpha_i^t, \ldots, \alpha_{|W|}^t)^T$
(24) $(B^t)^T = (\beta_1^t, \beta_2^t, \ldots, \beta_i^t, \ldots, \beta_{|W|}^t)^T$
(25) send $\mathrm{EDB}^t = (t, B^t, C_{F^t}, Q^t)$ to cloud server
(26) else if
(27) forming the ΔF
(28) for each file $f \in \Delta F$
(29) if $\mathrm{op}(f) =$ add or modify
(30) $f \longleftarrow$ the new f
(31) if $\mathrm{op}(f) = \mathrm{del}$
(32) $r \longleftarrow_R \{0,1\}^*$
(33) $f \longleftarrow r$
(34) (reorder)
(35) $R^{t+1} \longleftarrow (\{R\}, t+1)$
(36) $F^{t+1} \longleftarrow F' \bigcup \Delta F$
(37) for each file $f \in F^{t+1}$
(38) $c^{\mathrm{t+1}} \longleftarrow R^{\mathrm{t+1}}(f)$
(39) (generate the new index)
(40) for each $w \in \Delta W$
(41) if the $\mathrm{op}(w) =$ add
(42) $W^{\mathrm{t+1}} \longleftarrow w \bigcup W$
(43) $k_w \longleftarrow P_1(\mathrm{WEK}, H_1(w))$
(44) $\mathrm{ek}_w \longleftarrow P_2(\mathrm{WEK}, H_1(w))$
(45) if the $\mathrm{op}(w) = \mathrm{del}$
(46) $W^{\mathrm{t+1}} \longleftarrow W - w$
(47) $Q^{t+1} \longleftarrow (\{\mathrm{Q}\}, t+1)$
(48) for each $w \in W^{t+1}$
(49) $\alpha_w^{t+1} = (a_{w1}, a_{w2}, \ldots, a_{wl})$
(50) $\sigma_w^{t+1} \longleftarrow \mathrm{HMAC.Auth}(k_w, \tau, \alpha_w^{t+1})$
(51) $\beta_w^{t+1} \longleftarrow \mathrm{Enc}_{\sigma_w^{t+1}}(\alpha_w^{t+1})$
(52) $i^{t+1} \longleftarrow Q^{t+1}(\sigma_w^{t+1})$

Algorithm 1: Continued.

(53) $\Delta\alpha_w^t \longleftarrow \alpha_w^{t+1} - \alpha_w^t$
(54) $\Delta\sigma_w^t \longleftarrow \text{HMAC.Auth}(k_w, \tau, \Delta\alpha_w^t)$
(55) $(A^t)^T = (\alpha_1^t, \alpha_2^t, \ldots, \alpha_i^t, \ldots, \alpha_{|W^{t+1}|}^t)^T$
(56) $(B^t)^T = (\beta_1^t, \beta_2^t, \ldots, \beta_i^t, \ldots, \beta_{|W^{t+1}|}^t)^T$
(57) $\text{EDB}^{t+1} \longleftarrow (t+1, B^{t+1}, C_{F^{t+1}}, Q^{t+1})$
(58) send EDB^{t+1} to cloud server
(59) send $\Delta\lambda_w^t = \{(w, \Delta\sigma_w^t)\}$ to users

Search: $(t, UL, w, EDB^t) \longrightarrow C_w^t$

user

(1) if there is no updation, then
(2) $\sigma_w^t \longleftarrow (t, w, \lambda_w^t)$
(3) $\text{Tr}_u(w) \longleftarrow \text{PK.Enc}(\sigma_w^t)$
(4) else
(5) $ek_w \longleftarrow P_2(\text{WEK}, H_1(w))$
(6) $\sigma_w^{t+1} \longleftarrow \text{HMAC.Eval}(ek_w, \sigma_w^t, \Delta\sigma_w^t)$
(7) $\text{Tr}_u(w) \longleftarrow \text{PK.Enc}(\sigma_w^{t+1})$
(8) sends$\text{Tr}_u(w)$ to cloud server

cloud server:

(9) if u cannot be found in UL, then
(10) output error
(11) else
(12) $\sigma_w^t \longleftarrow \text{PK.Dec}(\text{Tr}_u(w))$
(13) $i \longleftarrow Q^t(\sigma_w^t)$
(14) $\alpha_i^t \longleftarrow \text{Dec}_{\sigma_w^t}(\beta_i^t)$
(15) if $a_{ic} = 1 (0 \le c \le l)$
(16) $C_w^t \longleftarrow C_w^t \bigcup (c, C_c^t)$
(17) returns C_w^t and β_i^t to user

Algorithm 1: FBM-DSSE.

Adduser: $(u, \text{SK}) \longrightarrow$ UL and USK
(1) $\text{UL} \longleftarrow \text{UL} \bigcup \{u\}$
(2) $\text{USK} \longleftarrow (\text{WEK}, P_2, H_1, \text{FK}, \lambda_w^t = \{(w, \sigma_w^t)\})$
(3) sends UL = $\{u\}$ to cloud server

Algorithm 2: Adduser.

Decrypt: $(C_w^t, H, FK) \longrightarrow F_w$
(1) $k_c \longleftarrow H(FK, c)$
(2) $f_c^t \longleftarrow \text{Dec}_{k_c}(C_c^t)$

Algorithm 3: Decrypt.

Verify: $(w, \beta_i^t, (\alpha_i^t)', \sigma_w^t) \longrightarrow$ reject or accept
(1) transform β_i^t to $(\alpha_i^t)'$ and send $(w, \beta_i^t, (\alpha_i^t)', \sigma_w^t)$ to data owner
(2) if $0 \longleftarrow \text{HMAC.Ver}(k_w, (\alpha_i^t)', \mathcal{P}, \sigma_w^{t+1})$
(3) returns "reject"
(4) if $1 \longleftarrow \text{HMAC.Ver}(k_i, \sigma_i^\tau, (\alpha_i^t)', \mathcal{P}, \sigma_w^{t+1})$
(5) returns "accept"

Algorithm 4: Verify.

cloud server being semihonest, the cloud sever will search 1 in the index and return corresponding files, so the returned file does contain the keyword w.

5.3.2. Security analysis. Then, we will show the security analysis of the proposed scheme.

Theorem 1 (adaptive security of FBM-DSSE). *Let P_1, P_2, H be secure PRF, $\Pi_1 = (Setup, Enc, Dec)$ be a CPA-secure symmetric encryption, and $\Pi_2 = (HMAC.Setup, HMAC.Auth, HMAC.Ver, HMAC.Eval)$ be a secure homomorphic message authenticator scheme. We define that*

$$\mathcal{L}_{\text{FBM-DSSE}} = \left(\mathcal{L}_{\text{FBM-DSSE}}^{\text{Search}}, \mathcal{L}_{\text{FBM-DSSE}}^{\text{Update}}\right), \tag{6}$$

where $\mathcal{L}_{FBM-DSSE}^{Search}(w) = (Time(w), rp(w), sp(w))$ and $\mathcal{L}_{FBM-DSSE}^{Update} = bs$. Then, FBM-DSSE is $\mathcal{L}_{FBM-DSSE}$ -adaptively secure.

Proof. As mentioned above, the server is the semihonest adversary $\mathcal{A}$ who correctly follows the protocol but attempts to use the messages received to learn information that should remain private. The challenger $\mathcal{C}$ is responsible for generating EDB and the search tokens of w. The simulator $\mathcal{S}$

simulates the view between $\mathcal{A}$ and $\mathcal{C}$ according to the leakage functions $\mathcal{L}_{\mathrm{FBM-DSSE}}$.

Game G_0: G_0 is the same as the real world game $\mathrm{DSSEREAL}_{\mathcal{A}}^{\mathrm{FBM-DSSE}}(\lambda)$, and there is

$$\Pr\left[\mathrm{DSSEREAL}_{\mathcal{A}}^{\mathrm{FBM-DSSE}}(\lambda)=1\right]=\Pr[G_0=1]. \quad (7)$$

Game G_1: in G_1, when querying H to generate the key for a column c, the challenger $\mathcal{C}$ chooses a new random key if the column c is never queried before and stores it in a table Key. Otherwise, return the key corresponding to w in the table Key. The adversary $\mathcal{A}$ cannot be able to distinguish between G_0 and G_1, otherwise we can build an adversary $\mathcal{B}_1$ to distinguish between H and a truly random function. More formally,

$$\left|\Pr[G_0=1]-\Pr[G_1=1]\right|\le \mathrm{Adv}_{F,\mathcal{B}_1}^{\mathrm{PRF}}(\lambda). \quad (8)$$

Game G_2: in G_2, we model the R as a table just like H in G_1; then, we can build an adversary $\mathcal{B}_2$ to distinguish between R and a truly random function, and there is

$$\left|\Pr[G_1=1]-\Pr[G_2=1]\right|\le \mathrm{Adv}_{F,\mathcal{B}_2}^{\mathrm{PRF}}(\lambda). \quad (9)$$

Similarly, we build the G_3 forP_1 and G_4 for P_2.

Game G_5: in G_5, as shown in Algorithm 1, in the update stage, the challenger $\mathcal{C}$ randomly picks a string for every keyword w as the updated index and gets the new EDB. Now, we will show that the adversary $\mathcal{A}$ gets more things than the new EDB is negligible.

(1) In the update stage, $\mathcal{Q}$ changed the order of the keywords in the new index. The two search tokens were σ_w^t and σ_w^{t+1}. The adversary could not get specific information about the keywords based on them. Even if the index corresponding to the keyword has not changed, HMAC.Auth will generate a corresponding label for a string of 0. Otherwise, it would conflict with the authentication of HMAC.

(2) The probability of using the search token corresponding to the keyword w to decrypt the indexes of other keywords w is negligible. Even if the indexes are the same, because the difference among keywords decides that in corresponding keys, it is impossible to achieve $\mathrm{Dec}_{\sigma_w^t}(\mathrm{Enc}_{\sigma_w^t}(\alpha_{w'}^t))$, where $w'\neq w$, otherwise it will contradict the security of symmetric encryption.

(3) Similarly, even if the file has not changed and the corresponding encryption key has not changed, it is impossible for an adversary $\mathcal{A}$ to obtain relevant information about the file based on the ciphertext of the index because symmetric encryption is CPA-secure.

(4) Obviously, for an adversary $\mathcal{A}$ who does not have the ek, even if he obtains $\Delta\sigma_w^t$, he will not be able to obtain a new key. Otherwise, there will be an adversary that can break the security of HMAC.

Simulator. We can replace the searched keyword w with $\mathrm{sp}(w)$ in G_5 to simulate the simulator $\mathcal{S}$. And now we are ready to show that G_5 and Simulator $\mathcal{S}$ are indistinguishable. For update, it is obvious since we choose new index for each update in G_5. For search, $\mathcal{S}$ chooses a new search token according to the $\mathcal{L}_{\mathrm{FBM-DSSE}}^{\mathrm{Search}}(w)$ which can be modeled by tables and does the encryption. So,

$$\Pr[G_5=1]=\Pr\left[\mathrm{DSSEIDEAL}_{\mathcal{A},\mathcal{S}}^{\mathrm{FBM-DSSE}}(\lambda)=1\right]. \quad (10)$$

Finally,

$$\left|\Pr\left[\mathrm{DSSEREAL}_{\mathcal{A}}^{\mathrm{FBM-DSSE}}(\lambda)=1\right]-\Pr\left[\mathrm{DSSEIDEAL}_{\mathcal{A},\mathcal{S}}^{\mathrm{FBM-DSSE}}(\lambda)=1\right]\right|\le\varepsilon, \quad (11)$$

where ε is negligible, which completes the proof. □

Corollary 1 (adaptive forward privacy of FBM-DSSE). *FBM-DSSE is forward private.*

Proof. From Theorem 1, we can infer that FBM-DSSE achieves forward privacy because the leakage function $\mathcal{L}_{\mathrm{FBM-DSSE}}^{\mathrm{Update}}(w)$ of FBM-DSSE does not leak more information than that defined in Definition 3. □

Corollary 2 (adaptive Type – I$^-$ backward privacy of FBM-DSSE). *FBM-DSSE is Type – I$^-$ backward private.*

Proof. From Theorem 1, we can get the conclusion that FBM-DSSE does achieve Type–I backward privacy since the leakage functions of FB-DSSE only leaks the same information as defined in Definition 4. □

6. Multifunctional Search

Due to using the bitmap index, our scheme is easily extended to support multifunctional search. Specifically, Hu et al. [29] proposed an efficient and secure multifunctional searchable symmetric encryption schemes which supports wildcard search, similarity search (including hamming distance and edit distance), fuzzy keyword search, and disjunctive keyword search simultaneously. Hu's scheme builds a bloom filter for every keyword followed by a encryption index $I' = Enc(k_s, F_w, w, r\,d)$, where k_s is the symmetric private key and $r\,d$ is the random value generated by a random function. If one replaces the encryption index in [29] with our bitmap index, the new scheme can achieve all the functional searches in [29].

As for the conjunctive keyword search, the primary schemes first search for one keyword at a time and then collect the results together. Our scheme can achieve a more effective way. First, data users provide keywords' search token to the server; then, the server decrypts the

Table 2: Comparison with prior forward private SSE schemes.

Schemes	Security	Dynamism	Multiuser	Computable Complexity			Storage	Communication complexity
	F/B/CKA	D/S	S/M	Cloud	R	Update	Do	Update do to C
Stefanov et al. [18]	F	D	S	$O(\min\{a_w + \log N, n_w \log^3 N\})$	1	$O(\log^2 N)$	$O(N^\alpha)$	$O(\log N)$
$\sum o\phi o\varsigma$ [19]	F	D	S	$O(a_w + d_w)$	1	$O(1)$	$O(W \log D)$	$O(1)$
Wang et al. [20]	F	D	M	$O(\lvert D(w)\rvert)$	2	$O(1)$ or $O(\lvert D(w)\rvert)$	$O(1)$	$O(1)$
Khons et al. [23]	$F + B3$	D	S	$O(\lvert D(w)\rvert)$	2	$O(1)$	$O(m \log D + D \log W)$	$O(1)$
Zuo et al. [26]	$F + B1^-$	D	S	$O(a_w)t_{\text{ma}}$	1	$O(1)t_{\text{ma}}$	$O(W \log D)$	$O(W)$
SSE-1 [31]	NA	S	S	$O(\lvert D(w)\rvert)$	1	-	$O(1)$	-
SSE-2 [31]	A	S	S	$O(\lvert D(w)\rvert)$	1	-	$O(1)$	-
M-SSE [31]	NA	S	M	$O(\lvert D(w)\rvert)$	1	-	$O(1)$	-
Bost [35]	$F + B1$	D	S	$O(a_w \log N + \log^3 N)$	3	$O(\log^2 N)$	$O(1)$	$O(\log^3 N)$
FBM-DSSE	$F + B1$	D	M	$O(1)$	1	$O(u_w + D)$	$O(1)$	$O(W)$

corresponding rows in the bitmap index, conducts the conjunctive operation, and finally returns the result set.

7. Performance Analysis

In this section, we first summarize the comparisons between our scheme and prior forward private schemes. See Table 2, where F/B in security, respectively, shows the forward privacy/backward privacy, where B1 means Type – I backward privacy, $B1^{-}$ describes Type – I^{-} backward privacy, and B3 indicates Type – III backward privacy. NA/A means the scheme is nonadaptive/adaptive indistinguishability security; D/S in dynamism column describes dynamic or static; S/M shows single/multi; R in the column of computable complexity denotes the round in search phase; Do represents the date owner; D denotes the number of documents in the document collection; N is the number of keyword/file-identifier pairs; a_w/d_w is the number of added/deleted entries for keyword w; W is the collection of distinct keywords; t_{ma} is the computational time of a modular addition; $D(w)$ is the number of files currently matching keyword w; m means the number of subkeywords; and u_w is the number of the keywords whose $D(w)$ has changed.

Then, we evaluate the performance of the proposed scheme. We implement our scheme using LINUX in Windows 10 with an Intel Core i7-8550U CPU 2.00 GHz processor and 16 GB memory. We simulate the update algorithm and search algorithm on this machine to evaluate the computation overhead time cost of our scheme. In our experiment, the size of each file is 50 kB, the number of the keywords is 5,000, and the symmetric encryption scheme and public key encryption scheme are AES and RSA, respectively. We simulate the number of files on 10,000 to 100,000 by an increase of 10,000 each time.

In the update phase, the data owner needs l times (number of columns) symmetric encryption operations, $|W|$ times index homomorphic message authentication, and $|W|$ times symmetric encryption; Figure 2 shows the total time required to generate the index.

In the search phase, the user only needs to perform the public key encryption operation once, and if there is an update, he needs to perform another homomorphic operation (when only one keyword is searched); Figures 3 and 4 illustrate the time cost of search token generation for users and search time for the cloud server.

In the search phase, the server needs to perform a public key decryption operation and a symmetric decryption operation. Figure 5 displays that the time cost of search for server which is mainly in the RSA.

These figures illustrate that the effect of the increase in the number of files on the search phase is linear, which is reasonable but an exponential growth for index generation. The crux of our scheme is the update algorithm, so we concluded that the proposed scheme is more suitable for small databases and achieves the enhanced security guarantees at the reasonable price of efficiency.

In order to simplify the update algorithm, we propose two more efficient methods for updating at the cost of losing some security. It is a tradeoff between efficiency and security.

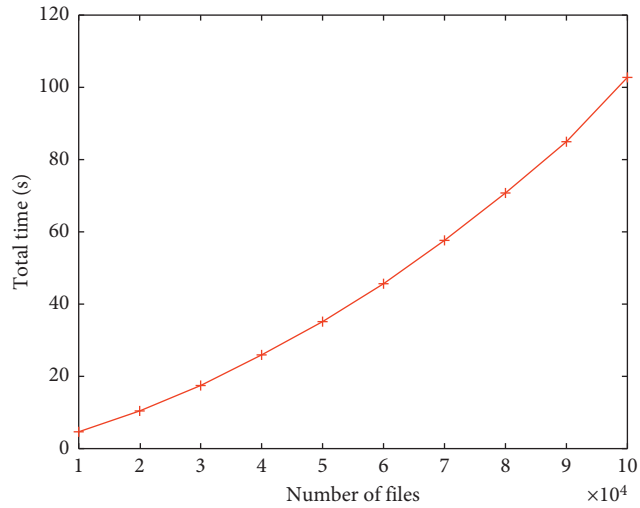

Figure 2: Index generation time.

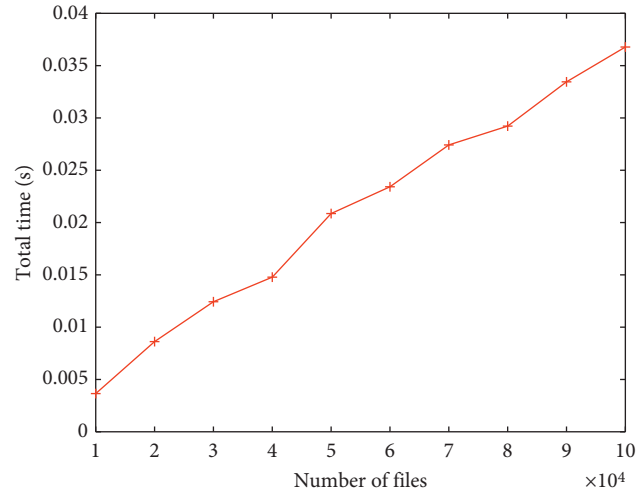

Figure 3: Search token generation time.

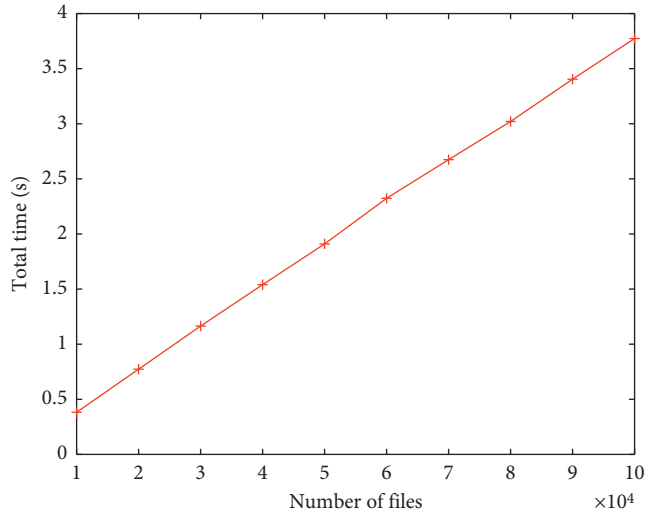

Figure 4: Search time.

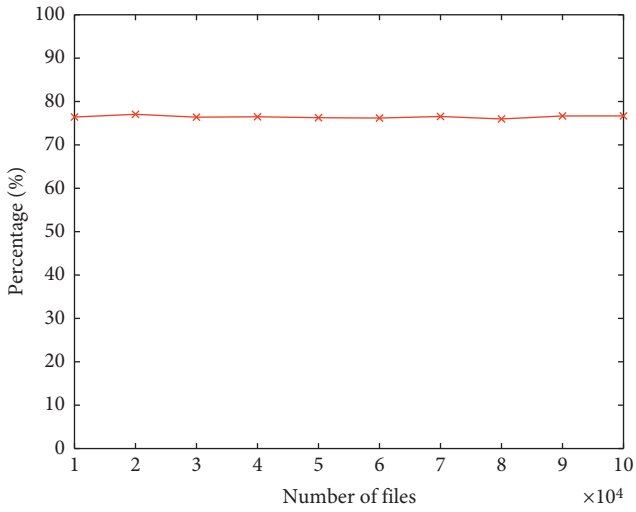

Figure 5: The proportion of RSA time search process.

One is a new update algorithm named the cycle Δt. For files that will not be updated in the short term, the data owner calculates the ciphertext of time t to time $t + \Delta t$ and Δt is a cycle. When the data owner is updated at time t, he arranges them in the corresponding column in chronological order. If the update time is $t' \in \{t, t + \Delta t\}$, the data owner only needs to upload the files that need to be changed and a certain amount of dummy data without downloading the files that does not change. Taking $t = 3$ as an example, the specific description is as follows.

Suppose there are f_1, f_2, f_3, f_4, f_5 and the time periods are, respectively, $t, t + 1, t + 2, t + 3$. Call R in advance from t to $t + 3$ and they are mapped to $(1, 2, 3, 4, 5)$, $(3, 4, 5, 1, 2)$, $(2, 3, 4, 5, 1)$, $(4, 5, 1, 2, 3)$. When the data owner updates data at time t, he encrypts the f_1 with $k_{c1}, k_{c4}, k_{c5}, k_{c3}$ (where k_{ci} means the i column's key), the f_2 with $k_{c2}, k_{c5}, k_{c1}, k_{c4}$, the f_3 with $k_{c3}, k_{c1}, k_{c2}, k_{c5}$, the f_4 with $k_{c4}, k_{c2}, k_{c3}, k_{c1}$, and the f_5 with $k_{c5}, k_{c3}, k_{c4}, k_{c2}$ and sends

$$\begin{aligned}
c_1&: E(k_{c1}, f_1) \| E(k_{c1}, f_3) \| \| E(k_{c1}, f_2) \| E(k_{c1}, f_4),\\
c_2&: E(k_{c2}, f_2) \| E(k_{c2}, f_4) \| \| E(k_{c2}, f_3) \| E(k_{c2}, f_5),\\
c_3&: E(k_{c3}, f_3) \| E(k_{c3}, f_5) \| \| E(k_{c3}, f_4) \| E(k_{c3}, f_1),\\
c_4&: E(k_{c4}, f_4) \| E(k_{c4}, f_1) \| \| E(k_{c4}, f_5) \| E(k_{c4}, f_2),\\
c_5&: E(k_{c5}, f_5) \| E(k_{c5}, f_2) \| \| E(k_{c5}, f_1) \| E(k_{c5}, f_3),
\end{aligned} \tag{12}$$

to the server.

When the user is searching at time t and the search result is $\{f_1, f_3, f_5\}$, the server just returns $E(k_{c1}, f_1), E(k_{c3}, f_3), E(k_{c5}, f_5)$. This new update algorithm cycleΔt can efficiently reduce the communication complexity of updates during the period.

Another method is that the data owner can use a proxy server. The proxy server is semihonest in the sense that it honestly runs the protocols but is curious to obtain privacy information. Additionally, it cannot collude with the cloud server. The proxy server is mainly responsible for the update of the database. Specifically, the data owner calculates the updated ciphertext together with the proxy server and stores it on the proxy server. And then proxy server interacts with the cloud server according to the specified time period and updates the ciphertext. Moreover, it is also responsible for updating the trapdoor. And we are studying further how to construct a FBM-DSSE scheme under a proxy server or a malicious server in the future.

8. Conclusions

In this work, we propose a Dynamic Searchable Symmetric encryption scheme for multiuser with Forward and Backward Security (FBM-DSSE). The proposed scheme realizes the Forward and Backward Security in Dynamic Searchable Symmetric encryption for multiuser. More specifically, the proposed scheme adopts a keyed pseudorandom function to hide the correspondence between files and indexes, takes symmetric encryption to improve the efficiency of file encryption and update, and uses HMAC to improve the efficiency of updating search tokens as files are changed. Furthermore, our scheme also supports verifiability and can be extended to multifunctional search. Further research work aims to reduce the computational complexity and communication complexity of the data owner and server. It will also combine access control to achieve fine-grained user management and file search.

Data Availability

All data included in this study are available upon request from the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61772311 and 62072276).

References

[1] D. X. Song, D. A. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," *IEEE Symposium on Security and Privacy*, vol. 3, no. 3, pp. 44-45, 2000.

[2] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," *Advances in Cryptology-EUROCRYPT 2004*, vol. 26, pp. 506–522, 2004.

[3] E. Goh, "Secure indexes," *IACR Cryptology ePrint Archive*, vol. 216, 2003.

[4] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proceedings of the Computational Science and Its Applications – ICCSA 2020*, pp. 1249–1259, Cagliari, Italy, July 2008.

[5] B. Zhu, J. Sun, J. Qin, and J. Ma, "A secure data sharing scheme with designated server," *Security and Communication Networks*, vol. 19, pp. 1–16, 2019.

[6] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, 2019.

[7] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 965–976, Raleigh, NC, USA, October 2012.

[8] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," *Financial Cryptography and Data Security*, vol. 25, 2013.

[9] M. Naveed, M. Prabhakaran, and C. A. Gunter, "Dynamic searchable encryption via blind storage," *IEEE Symposium on Security and Privacy*, vol. 2, no. 5, pp. 639–654, 2014.

[10] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[11] C. Guo, X. Chen, Y. M. Jie et al., "Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption," *IEEE Transations on Services Computing*, vol. 99, p. 1, 2017.

[12] Y. Miao, R. Deng, K.-K. R. Choo, X. Liu, J. Ning, and H. Li, "Optimized verifiable fine-grained keyword search in dynamic multi-owner settings," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 99, p. 1, 2019.

[13] M. S. Nair and M. S. Rajasree, "Fine-grained search and access control in multi-user searchable encryption without shared keys," *Journal of Information Security and Applications*, vol. 41, pp. 124–133, 2018.

[14] Y. B. Miao, X. Liu, K. K. R. Choo et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, p. 1, 2019.

[15] Y. Yang, X. Liu, and R. Deng, "Multi-user multi-keyword rank search over encrypted data in arbitrary language," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 99, p. 1, 2017.

[16] J. K. Liu and R. Steinfeld, "Multi-user cloud-based secure keyword search," *ACISP*, vol. 1, pp. 227–247, 2017.

[17] Y. P. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: the power of file-injection attacks on searchable encryption," *IACR Cryptology ePrint Archive*, vol. 172, 2016.

[18] E. Stefanov, C. Papamanthou, and E. Shi, *Practical Dynamic Searchable Encryption with Small Leakage*, National Down Syndrome Society, New York, NY, USA, 2014.

[19] R. Bost, "$\sum o\phi o\varsigma$: forward secure searchable encryption," in *Proceedings of the ACM Conference on Computer and Communications Security*, Orlando, FL, USA, November 2016.

[20] Q. Wang, Y. Guo, H. J. Huang et al., "Multi-user forward secure dynamic searchable symmetric encryption," *NSS*, vol. 125, 2018.

[21] S. F. Sun, X. L. Yuan, K. Joseph et al., "Practical backward-secure searchable encryption from symmetric puncturable encryption," in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 763–780, Toronto, Canada, October 2018.

[22] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W. H. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," in *Proceedings of the CCS, ACM*, pp. 1449–1463, Dallas, TX USA, October, 2017.

[23] J. Li, Y. Huang, Y. Wei et al., "Searchable symmetric encryption with forward search privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, p. 1, 2019.

[24] D. Catalano and D. Fiore, "Practical homomorphic message authenticators for arithmetic circuits," *Journal of Cryptology*, vol. 31, no. 1, pp. 23–59, 2018.

[25] C. Guo, X. Fu, Y. Mao, G. Wu, F. Li, and T. Wu, "Multi-user searchable symmetric encryption with dynamic updates for cloud computing," *Information*, vol. 9, no. 10, p. 242, 2018.

[26] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, "Dynamic searchable symmetric encryption with forward and stronger backward privacy," *Lecture Notes in Computer Science*, vol. 2, pp. 283–303, 2019.

[27] J. Hua, Y. Liu, H. Chen, X. Tian, and C. Jin, "An enhanced wildcard-based fuzzy searching scheme in encrypted databases," *World Wide Web*, vol. 23, no. 3, pp. 2185–2214, 2020.

[28] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. J. Lou, "Enabling efficient fuzzy keyword search over encrypted data in cloud computing," *IACR Cryptology ePrint Archive*, vol. 2019, p. 593, 2009.

[29] C. Hu, L. Han, and S. M. Yiu, "Efficient and secure multi-functional searchable symmetric encryption schemes," *Security and Communication Networks*, vol. 9, no. 1, pp. 34–42, 2016.

[30] S. Q. Lai, S. Patranabis, A. Sakzad et al., "Result pattern hiding searchable encryption for conjunctive queries," in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 745–762, Toronto, Canada, October 2018.

[31] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[32] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner, "Rich queries on encrypted data: beyond exact matches," in *Proceedings of the Computer Security--ESORICS 2015 ESORICS*, pp. 123–145, Vienna, Austria, September 2015.

[33] R. Li and A. X. Liu, "Adaptively secure conjunctive query processing over encrypted data for cloud computing," in *Proceedings of the ICDE*, pp. 697–708, San Diego, CL, USA, April 2017.

[34] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis, "Practical private range search revisited," in *Proceedings of the SIGMOD, ACM*, pp. 185–198, San Francisco, CL, USA, June 2016.

[35] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitive," in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1465–1482, Dallas, TX, USA, November 2017.

[36] R. A. Popa and N. Zeldovich, "Multi-key searchable encryption," *IACR Cryptology ePrint Archive*, vol. 508, 2013.

[37] S. Jarecki, C. S. Jutla, H. Krawczyk et al., "Outsourced symmetric private information retrieval," in *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 875–888, Orlando, FL, USA, November 2013.