Hindawi
Mobile Information Systems
Volume 2021, Article ID 7003252, 14 pages
https://doi.org/10.1155/2021/7003252

Research Article

Edge-Cloud-Assisted Multiuser Forward Secure Searchable Encryption (EMFSSE) Scheme in the P2P Networking Environment

Yongli Tang, Jingran Li, Xixi Yan, and Qiang Zhao

School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, Henan, China

Correspondence should be addressed to Xixi Yan; yanxx@hpu.edu.cn

Received 13 July 2021; Accepted 26 August 2021; Published 20 September 2021

Academic Editor: Han Wang

P2P network enables users to share resources effectively. However, with the advent of the big data era, the sensitive data of users in P2P network are also increasing dramatically. In order to solve the contradiction between the huge amount of sensitive data and the limited local storage space, an increasing number of users choose to encrypt their sensitive data and store them in the cloud server. For the problem of the secure storage and flexible access of large amounts of user data in P2P networks, an edge-cloud-assisted multiuser forward secure searchable encryption scheme is proposed. The scheme uses the proxy reencryption technique to optimize the multiuser searchable encryption and prevent the decryption key from being directly transmitted between users. By introducing an edge-cloud architecture, the system achieves efficient communication and timely response capabilities. The security analysis proves that our scheme achieves the CPA (chosen-plaintext attack) security based on DBDH assumption and the forward privacy. Finally, the theoretical and experimental comparisons between this scheme and other schemes show that our scheme has high efficiency in the process of data update, search, and trapdoor generation. In addition, due to the use of edge-cloud architecture, our scheme reduces about 90% and 75% of the user's consumption in the encryption and token generation process.

1. Introduction

P2P network file sharing system has been widely used and has rapidly developed because of its various advantages, such as decentralization, good extendibility, strong robustness, high cost-effectiveness, and load balancing characteristics [1–3]. With the development of communication and computer technology [4–6], the amount of data owned by users is increasing, but the local storage space of users is limited. This contradiction has become one of the main reasons for restricting the development of P2P networks. The development of cloud storage [7] has brought about a turning point for this situation. Clients can choose to transfer a large amount of local data [8] to a cloud server and use the cloud server to store, manage, and utilize the data. In addition, searchable encryption technology can ensure data security and flexible access during remote storage, which not only reduces local storage and computational cost but also achieves more flexible and convenient data access. However, in reality, there is a huge physical and logical distance between the client and the cloud server, which reduces the efficiency of data collection, transmission, and analysis, causes the delay of the communication process, and hinders the timely response ability of the whole system. This brings new challenges to the combination of file transmission and searchable encryption technology in P2P networking environment. A lot of work has shown [9–11] that edge-cloud collaboration can successfully replace the only-cloud system and strike a balance between powerful computing power and rapid response. The edge-cloud architecture is an optimized cloud computing system that transforms cloud service from the "cloud" as the center to the Internet edge and cloud cooperative operation, which overcomes the problems of only-cloud computing such as distributed data collection, transmission and response delay, and poor data analysis efficiency. With the help of edge computing, clients store some data information on the nearest edge platform, and the edge platform retrieves the data according to authorization, thereby limiting unnecessary data upload to the cloud, saving communication costs, and providing more efficient

and timely data search. In addition, the edge platform can also assist clients in some expensive operations and use the efficiency of edge computing to reduce the amount of calculation on the client side. Therefore, the edge-cloud architecture and the P2P network can be combined. Clients upload part of their data information to the edge-cloud architecture and enjoy the ability to share and use data in time.

The combined use of searchable encryption and edge-cloud collaboration can largely solve the contradiction between data storage, data confidentiality, and data transmission efficiency. It ensures that neither the edge side nor the cloud side can learn any useful information without completely sacrificing searchability of the data or that search function can be improved with minimal disclosure. However, the search server is generally assumed to be an honest but curious entity in a searchable encryption scheme. In the real world, the malicious server will carry out illegal operation on its stored ciphertext. Attackers can even use a small amount of leakage to leak the user's query. Particularly in dynamic databases, the attacker can inject new documents into the cloud server to gain more advantages, and the impact of file injection attacks can even be devastating. The searchable encryption scheme that realizes forward privacy can effectively resist such attacks [12]. Therefore, from the perspective of usage, it is very necessary to construct a forward secure searchable encryption scheme to solve the information security and privacy issues in the process of file updating. How to design a searchable encryption scheme with forward security under the edge-cloud architecture and apply it to P2P networks is the first goal of this paper.

In the P2P network, a node can not only be the data user who sends the request to obtain the required information but also provide the data as the data owner. Specifically, the P2P network aims to share various services among users. But so far, only a few schemes extend forward secure searchable encryption scheme to multiuser [13, 14]. Wang et al. [13] introduce a semihonest proxy server that does not collude with the cloud server to solve the user query problem. Lu et al. [14] improve Wang's scheme, remove the state value transfer between the data owner and the proxy server, and add the user authentication function. However, both schemes require the data owner to send the decryption key to the data user directly. In order to prevent the disclosure of the decryption key, this paper introduces the proxy reencryption technique. The authorized proxy (obtains the reencryption key generated by the data owner) transforms the ciphertext provided by the data owner into another ciphertext that can be decrypted by the data user. Then, the data user can download the ciphertext alone and decrypt it with his own private key to get the original plaintext. How to combine forward secure searchable encryption with proxy reencryption to provide secure file sharing between users is the second goal of this paper.

1.1. Contribution. In order to achieve the two goals mentioned above, we study a series of existing schemes. First of all, users in P2P network upload a large amount of data to the cloud for storage and search. However, the distance between clients and server causes serious operation delay in this process, which affects the performance of end-to-end communication in the P2P network. Inspired by the work [15], edge-cloud architecture can effectively realize timely response between client and edge platform and greatly improve communication efficiency. Secondly, malicious attackers may carry out a variety of attacks to damage the confidentiality of user data by using the information leaked in the communication process, for example, file injection attacks. Therefore, constructing a searchable encryption scheme with forward security is particularly important, which can effectively ensure that the file update process does not disclose data information. Finally, in order to combine P2P network with forward secure searchable encryption technology, it is necessary to implement forward secure searchable encryption technology among multiple users. However, the existing multiuser schemes need to transmit the decryption key directly [13, 14]. In order to prevent the decryption key from leaking, the proxy reencryption technique is introduced to optimize the security of file transmission of each node in the P2P network. In summary, we have constructed a multiuser forward secure searchable encryption scheme EMFSSE based on the edge-cloud architecture in the P2P network. In the scheme, we store part of the index information in the edge server nearest to the user and store encrypted files and encrypted indexes in the cloud server. The communication efficiency of the whole system is also enhanced because it can realize fast update, search, and transmission between the user and the server by making full use of the storage and computing capacity of cloud edge. The main contributions are as follows:

(1) This scheme combines the P2P network with the multiuser forward secure searchable encryption. It realizes the encryption storage, ciphertext search, and update function of user data in the P2P network environment. And it supports data transmission between users.

(2) Proxy reencryption technique is introduced to solve the decryption key leakage problem in the scheme [13, 14]. In our scheme, the edge server reencrypts the ciphertext, and then the data requester uses its own private key to decrypt the reencrypted ciphertext.

(3) We take advantage of edge-cloud collaborative computing to spread out the workload of only-cloud server and reduce the burden of users. Firstly, the edge server maintains part of the keyword information and assists the authorized data user to generate trapdoor. The data owner can authorize the data user to query the encrypted database dynamically and effectively. Secondly, because of the excellent computing power of edge server, it can be used to realize the reencryption process and convert part of the heavy encryption operations to the edge, which can ensure data confidentiality and reduce the computational cost of the client and cloud.

(4) The scheme satisfies the correctness and indistinguishable forward security under the adaptive attack model. And the scheme is proved to be against the chosen-plaintext attacks (CPA) based on the DBDH assumption during the reencryption process.

(5) Comparative analysis and experimental results show that the proposed scheme has a good efficiency in the process of encryption and searching. Compared with the scheme without edge computing, the proposed scheme can save about 92% of the computational costs of the client in the process of ciphertext generation. It can save about 75% of the computation cost in the token generation process.

1.2. Organization. The rest of this paper is organized as follows. In the next section, we introduce the related work. The description of symbols and related knowledge are introduced in Section 3. Section 4 analyzes the design goals and basic model. Then the specific structure and safety analysis of the EMFSSE scheme are introduced in Section 5 and Section 6, respectively. Section 7 compares the EMFSSE scheme with other related schemes and evaluates its performance. In the end, Section 8 summarizes the full text.

2. Related Work

This paper studies the multiuser forward secure searchable encryption technology optimized by proxy reencryption technique in the edge-cloud-assisted P2P network. Therefore, this section focuses on the research progress of forward secure searchable encryption, edge-cloud architecture, and proxy reencryption technique.

2.1. Forward Secure Searchable Encryption (FSSE). The desired goal of forward secure searchable encryption (FSSE) is that the malicious server does not disclose any information of existing data in the database during the file update phase. The scheme of Chang Mitzenmacher [16] already supports forward security. This paper points out that when a file is updated, the newly added file may be related to the previously queried keywords, thereby leaking some information. Stefanov et al. [17] proposed the concept of FSSE and the concept of backward security. Subsequently, the first scheme with good communication costs and computational complexity was proposed by Bost [12], which hides the relationship between document identifiers and keywords through trapdoors replacement in the indexing process and does not disclose the information of previously queried files when new files are injected. Subsequently, scholars have successively proposed various forward secure searchable encryption schemes [18–20] on the basis of the $\sum_{o\phi o\varsigma}$ scheme. The application of the FSSE scheme has also become a research hotspot recently. Chen et al. [21] proposed a blockchain-based forward secure searchable encryption scheme and applied it to vehicle trading websites. The scheme uses smart contract to replace the cloud server for search. Wang et al. [22] proposed a verifiable forward security ranking search scheme in P2P network environment. The scheme realizes ranking search and achieves forward security by changing the file vector structure and using modular residual computation. However, most of these schemes cannot realize file transfer between clients. Wang et al. [13] proposed an FSSE scheme to support multiuser search, which realizes data access by using proxy server to maintain keyword state information. However, this scheme directly transmits decryption key through secure channel, which not only causes the problem of decryption key leakage but also requires the data owner to keep online during the whole search phase.

2.2. Edge-Cloud Architecture. It is an effective way to optimize cloud computing by using cloud-edge collaboration instead of only-cloud system, which has become a research trend in the near future. The edge platform is closer to the underlying data source and has the geographical advantage of being closer to the real clients, so it can effectively reduce the system delay. And edge server can replace the client side or cloud side to perform some complex operations, which can improve the efficiency of the client and cloud server, and provide on-demand services. Ren et al. [23] proposed that the cloud-edge cooperation method can effectively improve the delay performance. Zhang et al. [24] used edge-cloud cooperation to construct a multidevice management service system and applied it to the industrial Internet of things. The system improved the response speed of data collected by basic equipment to a certain extent and reduced the network bandwidth load pressure brought by data transmission. Mollah et al. [25] combined cloud-edge collaboration with public key encryption to construct a secure data sharing scheme, which supports searching on encrypted data (searchable encryption), uses cloud-edge collaboration to reduce computing and communication costs, and provides privacy and confidentiality functions for outsourced data. Wang et al. [15] constructed a lightweight scheme of edge-cloud-assisted public key searchable encryption, using lightweight cryptography and edge-cloud computing to reduce the encryption cost of infrastructure equipment by 70%. But the previous schemes are all applied to the industrial Internet of things, and few schemes combine edge computing with P2P network.

2.3. Proxy Reencryption. Proxy reencryption technique means a trusted third party that converts data encrypted by the data owner with his public key into ciphertext data that can be decrypted by the data user with his private key, so as to complete data sharing. It is important to note that, during this process, no plaintext information should be disclosed to the proxy. In 1998, proxy reencryption technique was proposed by Blaze et al. [26]. With the development of cloud storage, a large number of researchers focus on the combination of proxy reencryption technique and cloud storage to achieve the purpose of cloud data sharing [27–29]. Early proxy reencryption schemes have the problem of reencryption key abuse. In 2005, [30] introduced the concept of nontransferability, but it was not until 2015 that Guo et al. [31] completed the instantiation of this concept using

indistinguishable confusion and unforgeable authentication. On this basis, a public accountability proxy reencryption scheme was proposed by Guo et al. [32] in 2021. Different from the traditional reencryption key generation, the generation process of this scheme is noninteractive, which saves the communication cost, and the scheme can determine the operator of the decryption operation, which improves the security of the scheme.

3. Preliminaries

This section mainly introduces the symbols and encryption-related knowledge used in this paper.

3.1. Notations. The notation information of this paper is given in the Table 1

3.2. Cryptography Materials

3.2.1. Trapdoor One-Way Permutation Technique. We cite the definition of trapdoor one-way permutation in literature [33].

The group D based on trapdoor permutation family Π consists of three algorithms: Random generation algorithm (Gen): it inputs security parameter λ and outputs permutation description s and related trapdoor t. Evaluation algorithm (Eva): it inputs s and the original image $x \in D$ and outputs the image of x, $y \in D$. Inverse algorithm (Inv): it inputs s, trapdoor t, and y and output the preimage of y in the permutation. And the one-way permutation function has the following properties:

(i) For all $\text{Gen} \longrightarrow_R (s, t)$, the related $\text{Eva}(s, \cdot)$ should be a permutation of group D

(ii) For all $x \in D$, $\text{Inv}(s, t, \text{Eva}(s, x)) = x$

3.2.2. Bilinear Pairing. According to literature [34], we briefly introduce the definition of bilinear pairing. Let G_1 and G_2 be two cyclic groups of the same prime order $p \geq 2^\lambda$, g be the generator of G_1, and λ be the security parameters. Let e: $G_1 \times G_1 \longrightarrow G_2$ denote a map to satisfy the following properties:

(i) Bilinear: For any $x, y \in G_1$ and $a, b \in Z_p$, the equation $e(x^a, y^b) = e(x, y)^{ab}$ holds

(ii) Computable: for $x, y \in G_1$ that meets the condition, an algorithm can be found to calculate the value of $e(x, y)$ in polynomial time

(iii) Nondegenerate: $e(g, g) \neq 1$

3.2.3. Decisional Bilinear Diffie–Hellman (DBDH) Assumption. According to literature [35], the description of DBDH assumption is given. The definitions of groups G_1 and G_2 and map e: $G_1 \times G_1 \longrightarrow G_2$ are the same as above. For any adversary B, its advantages are defined as $\text{Adv}_B^{\text{DBDH}}(\lambda) = \left| \begin{matrix} \Pr[B(g, g^a, g^b, g^c, e(g, g)^{abc} = 1] \\ -\Pr[B(g, g^a, g^b, g^c, e(g, g)^z = 1] \end{matrix} \right|$, where $a, b, c, z \leftarrow_R Z_p$. We say that the DBDH assumption holds if, for any probability polynomial time (PPT) adversary B, its advantage $\text{Adv}_B^{\text{DBDH}}(\lambda)$ is negligible with security parameter λ.

3.2.4. Forward Privacy Security. This paper cites the definition of forward privacy technology in literature [12].

Common Leakage. We define the leak function $L = (L^{\text{Setup}}, L^{\text{Search}}, L^{\text{Update}})$, which represents the amount of information leaked to the adversary in the searchable encryption scheme. The leak function L treats the query list Q (including all queries) as status. List Q stores each query for keyword w, and its element is (i, w) or $(i, \mathbf{op}, \mathbf{in})$ for an update query $\mathbf{op}$ with input $\mathbf{in}$. The integer i represents the timestamp with the initial value of 0, which increases with the number of queries.

(i) Search pattern: $\text{sp}(x) = \{j\colon (j, x) \in Q\}$ (only matches search queries)

(ii) Query pattern: $\text{qp}(x) = \{j\colon (j, x) \in Q \text{ or } (j, \text{op}, \text{in}) \in Q \text{ and } x \text{ appears } in \ \mathbf{in}\}$

Forward Privacy. A L-adaptively secure searchable encryption scheme $\sum$ is forward private if the update leakage function L^{Update} can be expressed as

$$L^{\text{Update}}(\mathbf{op}, \mathbf{in}) = L'(\mathbf{op}, \{(\text{ind}_i, u_i)\}), \tag{1}$$

where $\{(\text{ind}_i, u_i)\}$ represents the modified document set and u_i represents the number of modified words.

3.2.5. Binary Index Tree. We use the binary index tree in literature [18] to store the encrypted file information, as shown in Figure 1.

Block. A block b is generated for each index (w, ind). The block b is divided into two parts, the block address b.id and the block data b.data, where b.data stores the data value b.value (the information of one index (w, ind) in this paper), the previous block key denoted by b.pre.key, and the previous block address b.pre.id. To sum up, $b = (b.\text{id}, b.\text{value}, b.\text{pre.key}, b.\text{pre.id})$.

Blocks Chain. Connect the blocks with the same keyword w into a chain C. Chain C represents the index list related to w, and its length is equal to the number of file indexes n_w containing w. The total number of chains is equal to the number of keywords $|W|$. There are three types of blocks on each chain C: the first block b.head, the last block b.tail, and the other blocks b.internal, where $b.\text{tail} = (b.\text{id}, b.\text{value}, \perp, \perp)$.

Binary Index Tree. Each node of the binary index tree T stores a block b. The node indexed by id is denoted by b.id, and the stored data is denoted by b.data. The value of the data block stores all indexes (w, ind) containing the keyword w. Finally, the tree T has a total of $|W|$ index lists (chains), including $|W| \cdot n_w$ nodes, and the depth is $\log(|W| \cdot n_w)$.

Table 1: Notations description.

Notation	Description
λ, μ	The security parameter
(q, p, G_1, G_2, e)	The bilinear parameters
G, F	The one-way permutation functions
H_1, H_2, H_3, H_4	The hash functions
$\sum$	The map stored on the data owner side
$\mathbf{T}$	The list stored on the edge server side
T	The tree stored on the cloud server side
(pk_i, sk_i)	i's public/private key
(fk_o, fk_o', wk)	Data owner's secret key
m	The plaintext
ind	The identifier of m
w	The keywords
W	The set of keywords associated with m
op	The addition/deletion operation
C	The ciphertext of m
$\mathrm{Tra}(w)$	The query trapdoor
$\mathrm{Tr}(w)$	The search trapdoor
b	A data block stored in the tree T is used to store index information
$(b.\mathrm{id}, b.\mathrm{key})$	An identifier/key pair matches the data block b
(qk_c, qk_u)	The query key generated by the data owner for a data user
$rk_{o \longrightarrow u}$	The reencryption key generated by the data owner for a data user
EDB	The encrypted database
$x \leftarrow_R X$	Uniformly and randomly select x from the finite set
$\|$	The concatenation operator
$\oplus$	The XOR operator
C	A chain
W	The keyword dictionary
$\|W\|$	The number of keywords in the keyword dictionary
n_w	The number of files contained in a keyword w

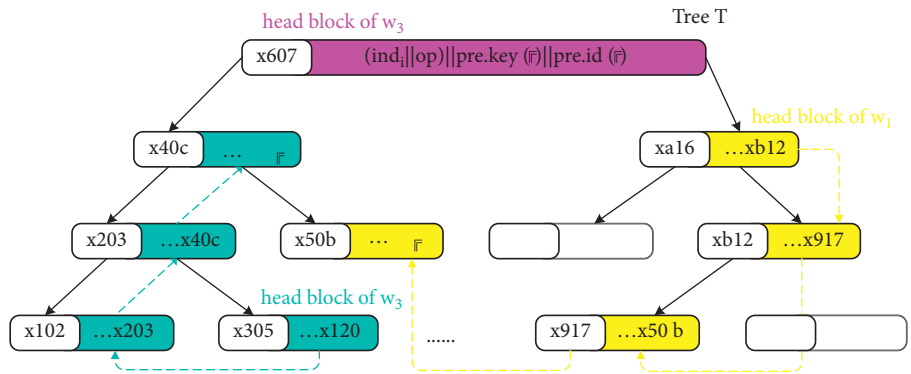

Figure 1: Binary index tree T (stores index information of three keywords (w_1, w_2, w_3)).

4. System Design

4.1. Design Goals. According to the information transmission requirements of P2P network and users, the scheme designed in this paper must meet the following goals:

Efficiency: the index structure in the searchable encryption scheme has an important impact on search efficiency. We store the index in a tree structure to improve search efficiency. In the phase of ciphertext generation and search query, we utilize the edge-cloud architecture to improve the fast response capability of the whole system.

Confidentiality: this is another key attribute. This scheme needs to ensure the confidentiality of data, index, and query. That is, the server cannot get the relevant information of the plaintext by analyzing the stored ciphertext or by reencrypting the ciphertext. And it also cannot get the relevant index information and data query when attacked by an adversary. In addition, this scheme provides forward privacy performance specifically against file injection attack [12].

Dynamic update: this scheme supports the dynamic update of files including add, delete, and update functions. When the user wants to modify the data, he

only needs to perform corresponding operations on the data index and the database.

Correctness: correctness is the basic property of a searchable decryption scheme, which requires the system to provide services correctly. Specifically, as long as the correct trapdoor is provided and the system is executed sequentially in a legal manner, the search results must be correct.

Multiusers: this attribute refers to the ability of the scheme to support data transmission between different users. And the data owner is able to authorize users and update their access rights at any time.

4.2. System Model. As shown in Figure 2, our scheme has four entities: data owner (DO), edge server (ES), cloud server (CS), and data user (DU). DO needs to outsource data while keeping the confidentiality and operation ability of data. At the same time, DO wants to have the right to control other users' access to the data. ES is responsible for storing part of the keyword information from the DO nearby, assisting the data user to generate trapdoor, and reencrypting the ciphertext found by CS. CS stores encrypted data, encrypts index, and responds to data queries. The scheme mainly includes the following processes:

(1) Step 1 (data update): DO first extracts index from the outsourced files and generates index information. He first uses a symmetric encryption algorithm to encrypt index and asymmetric encryption algorithm to encrypt documents. Then DO uploads part of the keyword information to ES and uploads the encrypted index and ciphertext to CS.

(2) Step 2 (user authorization). Whenever a new authorized user DU joins the system, the DO generates the query key and the reencryption key. DO sends the query key to DU and the reencryption key to the ES near the DU.

(3) Step 3 (search keyword). DU generates a query trapdoor containing the keyword. The query trapdoor is sent to the ES, and then the ES generates the trapdoor for search and sends it to the CS. CS decrypts the encrypted index from the search trapdoor to get the relevant document identifier and then obtains the required ciphertext from the searched file identifier. Finally, the CS sends the searched ciphertext to the ES near the DU.

(4) Step 4 (reencryption). After obtaining the matching ciphertext, the ES finds the stored reencryption key of the target user and reencrypts the ciphertext. Finally, the reencrypted ciphertext that the DU has the ability to decrypt is generated and sent to the DU.

(5) Step 5 (decryption). DU uses its private key to decrypt.

The basic framework of the FSSE scheme based on the edge-cloud architecture in P2P network EMFSSE is as follows:

(1) Setup (λ): it takes security parameters λ/μ as input and outputs public parameters *Para*.

(2) KeyGen (Para): enter the public parameter Para, and the ES and the DO generate the required key, respectively. Specifically, the ES generates its public/private key (pk_e/sk_e). The DO generates its public/private key (pk_o/sk_o), secret key of encryption index (fk_o/fk_o'), and keyword encryption key wk.

(3) Update (Para, m, ind, W, $\sum$): this operation is performed by the DO. It inputs public parameters Para, ciphertext m, document identifier ind, key set W related to the document, and keyword status map $\sum$ and outputs index information table T', encrypted index set **B**, and ciphertext C and updates the map $\sum$.

(4) Register (Para): when a new data user joins the system, first, his/her own public/private key (pk_u/sk_u) is generated. Then, DO authorizes the added user, inputs the public parameters*Para*, and generates the query key qk_u, query-related information qk_c, and reencryption key $rk_{o\longrightarrow u}$ for the new user DU. Finally, $(qk_c, rk_{o\longrightarrow u})$ is sent to the ES and qk_u is sent to the DU.

(5) Trapdoor (Para, qk_u, w): the DU takes public parameters Para, its query key qk_u, and query keyword w as input and outputs the trapdoor Tra (w) of w.

(6) Search (Para, Tra (w), **T**; T, EDB): this operation is performed by ES and CS in cooperation. ES inputs public parameters Para, trapdoor Tra (w), and index information table **T**, outputs query token token, and sends it to the CS. The CS inputs encrypted index tree T, encrypts database EDB, and finally outputs ciphertext set **C**.

(7) Re – Encryption $(rk_{o\longrightarrow u}, \mathbf{C})$: the ES inputs reencryption key $rk_{o\longrightarrow u}$ and ciphertext set **C** and outputs reencrypted ciphertext set C'.

(8) Decryption (C', sk_u): the operation is performed by the DU, which inputs the key sk_u and the reencrypted ciphertext set C' and outputs the plaintext m corresponding to each ciphertext C' in C'.

5. Scheme Construction

This section describes the structure of the scheme EMFSSE in detail.

5.1. System Initialization Phase. Setup $(\lambda, \mu) \longrightarrow$ (Para): taking security parameters λ and μ as input, the algorithm generates bilinear mapping e: $G_1 \times G_1 \longrightarrow G_2$, where G_1 and G_2 are two cyclic groups of order $p \geq 2^\lambda$. Then, it randomly chooses $g, g_1, g_2, h_1, h_2 \leftarrow_R G_1$, calculates $L = e(h_1, h_2)$, and lets $\mathbf{P} = (g, g_1, g_2, h_1, h_2, L)$; meanwhile, it selects some secure hash functions H_0: $\{0,1\}^* \longrightarrow \{0,1\}^{\lambda+\mu+1+\log N}$, $H_{1,1'}$: $\{0,1\}^* \longrightarrow \{0,1\}^\mu$, H_2: $\{0,1\}^* \longrightarrow \{0,1\}^\lambda$, and $H_{3,4}$: $\{0,1\}^* \longrightarrow \{0,1\}^*$ and selects one-way permutation functions G and F. Finally, the DO initializes an empty map $\sum$ that stores its state, the ES initializes the empty index information table **T**, and the CS

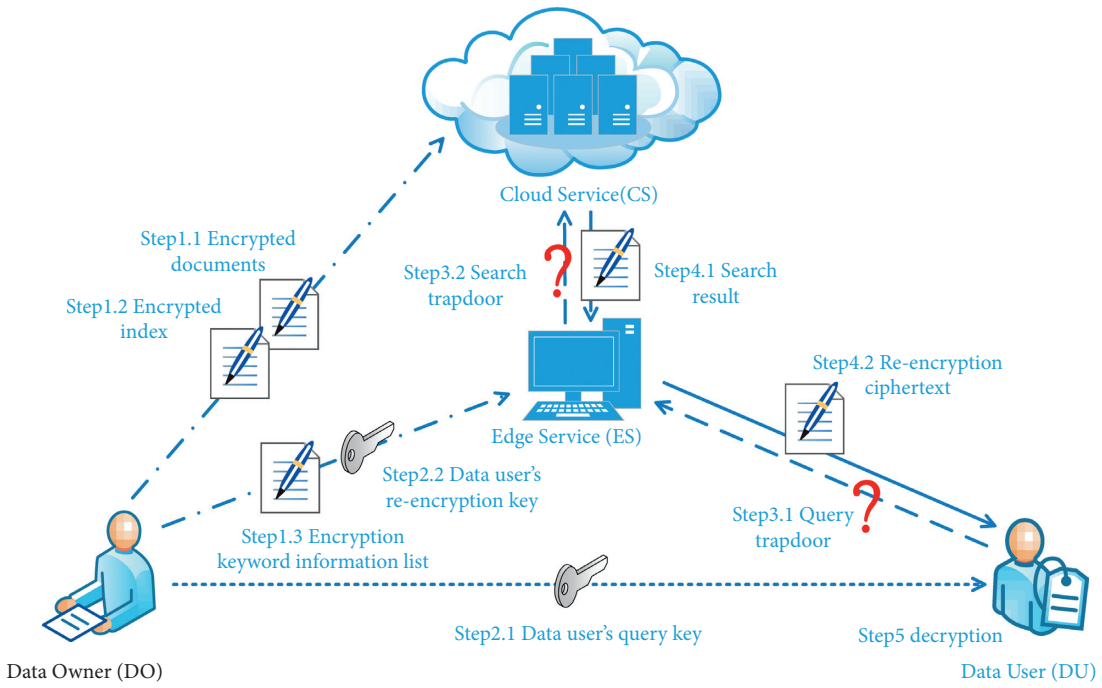

Figure 2: System model.

initializes an empty index tree T. Publish public parameters Para $= (\mathbf{P}, G, F, H_{0,1,1',2,3,4})$.

5.2. Key Generation Phase. KeyGen(Para) $\longrightarrow$ $(sk_e, pk_e; sk_o, pk_o, fk_o, fk_o', wk)$: it inputs the public parameters Para.

(1) Step 1: the ES randomly selects $z \leftarrow_R \mathbb{Z}_p$ and generates private key $sk_e = z$ and public key $pk_e = g_2^z$.

(2) Step 2: the DO randomly selects $x_o, y_o \leftarrow_R \mathbb{Z}_p$, respectively, generates private key $sk_o = (x_o, y_o)$ and public key $pk_o = (h_1^{x_o}, g_1^{y_o})$, respectively, selects key $fk_o, fk_o' \leftarrow_R \{0,1\}^\lambda$ of encryption index, and randomly selects key of encryption key $wk \leftarrow_R Z_p^*$.

5.3. Update Phase. Update(Para, m, op, ind, $W, \sum$) $\longrightarrow$ $(C, T', \mathbf{B})$: when a new file is added, the DO input public parameters. Para, file m, operator op = add (op = add/del, resp., for add or delete operation), file identifier ind, the file related keyword set W, and DO's status $\sum$, and then do operations as follows:

(1) Step 1 (documents encryption): the DO randomly selects $r \leftarrow_R Z_p$ and generates ciphertext $C = (L^r \cdot m, g_1^r, e(h_1, g_2)^{1/r}, x_o^r) = (c_0, c_1, c_2, c_3)$.

(2) Step 2 (index encryption): the DO initializes an empty set $\mathbf{B}$ and a table T'. For each keyword $w \in W$, DO first encrypts it to get $\mathrm{Tr}(w) = H_4(e(H_3(w), g)^{wk})$ and then uses function ReGen (as shown in Table 2) to calculate the identifier and key of the block currently indexed by keyword w, (id, key)$\leftarrow$ReGen$(w, \sum[w] = c)$. Then, DO makes pre.id = id, pre.key = key, and $c++$ and generates the identifier and key of the new block (id, key)$\leftarrow$ReGen(w, c). Finally, DO calculates the block key mask$\leftarrow H_0$(id, key), encrypts the index block with it $b \leftarrow$ (id, (//ind$\|$op//$\|$pre.key$\|$pre.id) $\oplus$mask), stores the index information (id, key) in the table T' indexed by $\mathrm{Tr}(w)$, where $T'[Tr(w)] =$ (id, key), and stores the block b in the set $\mathbf{B}$.

(a) The algorithm finally outputs index information table T', ciphertext C, and encrypted index set $\mathbf{B}$. The table T' is sent to the ES which stores its contents in an index information table. The ciphertext C and set $\mathbf{B}$ are sent to the CS which stores C in the encrypted database EDB and inserts the blocks into the encrypted index tree T.

Table 2: ReGen algorithm (note: i_w represents w or a unique identifier of w).

Algorithm ReGen$(w, \sum[w] = c)$
1: if $c == -1$ then id$\leftarrow\perp$, key$\leftarrow\perp$
2: else
3: $K_w \leftarrow F_{fk_o}(w), K_w' \leftarrow F_{fk_o'}(w)$
4: $ST_0(w) \leftarrow G(i_w)$
5: if $c == 0$ then
6: id$\leftarrow H_1(K_w, ST_0(w))$; key$\leftarrow H_2(K_w', ST_0(w))$
7: else
8: $ST_c(w) \leftarrow H_{1'}^c(ST_0(w))$
9: id$\leftarrow H_1(K_w, ST_0(w))$; key$\leftarrow H_2(K_w', ST_0(w))$
10: end if
11: end if
12: Return (id, key)

5.4. Authorization Phase. Register(Para) $\longrightarrow$ $(sk_u, pk_u; qk_u, qk_c, rk_{o\longrightarrow u})$: input the public parameter Para; the algorithm is completed by the new user DU and DO in order.

(1) Step 1: when a new user DU joins the system, it randomly selects $x_u, y_u \leftarrow_R \mathbb{Z}_p$ to generate a private key $sk_u = (x_u, y_u)$ and a public key $pk_u = (h_1^{x_u}, g_1^{y_u})$. Then, publish public key, indicating that a new user has joined the system.

(2) Step 2: the new user DU sends his public key to DO and then is authorized by DO to access the data. The DO randomly selects $qk_u \leftarrow_R Z_p^*$ as the query key, calculates $qk_c = g^{wk/qk_u}$, and generates the proxy reencryption key $rk_{o\longrightarrow u} = (h_2 g_1^{y^u} g_2^z)^{1/x_o}$ of DU.

5.5. Trapdoor Generation Phase. Trapdoor(Para, qk_u, w) $\longrightarrow$ Tra(w): the DU inputs the public parameter Para, the query key qk_u, and the query keyword w, then generates the query trapdoor $\mathrm{Tra}(w) = H_3(w)^{qk_u}$, and sends it to ES.

5.6. Search Phase. Search(Para, Tra(w), **T**; T, EDB) $\longrightarrow$ (**C**): the algorithm is completed by the cooperation of ES and CS.

(1) Step 1: input the query trapdoor Tra(w) and the encrypted keyword status table **T**, and the ES generates $\mathrm{Tr}(w) = H_4(e(\mathrm{Tra}(w), qk_c))$, then queries the table **T** to get the search trapdoor $(\mathrm{id}, \mathrm{key}) \leftarrow \mathbf{T}[\mathrm{Tr}(w)]$, and sends the search trapdoor token = (id, key) to the CS.

(2) Step 2: taking the search trapdoor token, index tree T, and encryption database EDB as input, the CS first initializes the empty sets **S** and **C** and sets $i \leftarrow 0$. Then, CS finds the latest block $b \leftarrow T[\mathrm{id}]$ by trapdoor token, calculates the block key mask$\leftarrow H_0$(key, id), decrypts the block with mask, and obtains $(\prime\prime \mathrm{ind} \| \mathrm{op} \prime\prime \| \mathrm{pre.key} \| \mathrm{pre.id}) \leftarrow b.\mathrm{data} \oplus \mathrm{mask}$, where b.value = (ind$\|$op). And the CS stores b.value in $\mathbf{S}[i \pm \pm]$ (merges the same ind in **S** according to op), sets id$\leftarrow$pre.id and key$\leftarrow$pre.key, and repeats the above process until id $\neq \perp$ and key $\neq \perp$. Finally, CS finds the corresponding encrypted data C in EDB by ind $\in$ **S**, stores C in **C**, and sends the search result **C** to the ES.

5.7. Reencryption Phase. Reencryption(**C**, $rk_{o\longrightarrow u}$) $\longrightarrow$ (C'): the ES finds the reencryption key $rk_{o\longrightarrow u}$ corresponding to the DU, inputs it and set **C**, and initializes an empty set C'. For each $C \in \mathbf{C}$, the ES reencrypts it to get reencrypted ciphertext $C\prime = (c_0', c_1', c_2') = (c_0, c_1, e(c_3, rk_{o\longrightarrow u})/c_2^z)$, stores the new ciphertext in C', and then sends C' to the DU.

5.8. Decryption Phase. Decryption(C', sk_u) $\longrightarrow$ (m): inputting reencrypted ciphertext C' and private key sk_u, DU decrypts each $C' \in C'$ to get all the required ciphertext $m = c_0' \cdot e(h_1, c_1')^{y_u} / c_2'$. The algorithm can get the final plaintext files m.

6. Security Proof

In this section, some important design objectives mentioned in Section 3 will be proved. Specifically, we will prove that the scheme can provide correctness, confidentiality, and forward security.

6.1. Correctness. An example is given to prove the correctness of the scheme. Suppose that there are files m_1, m_2 with keyword w_1, and their corresponding file identifiers are $\mathrm{ind}_1, \mathrm{ind}_2$.

Firstly, plaintext and index information are encrypted, respectively. We randomly select $r_1, r_2 \leftarrow_R Z_p$, generate ciphertexts $C_1 = (L^{r_1} \cdot m_1, g_1^{r_1}, e(h_1, g_2)^{1/r_1}, X_o^{r_1}) = (c_0^1, c_1^1, c_2^1, c_3^1)$ and $C_2 = (L^{r_2} \cdot m_2, g_1^{r_2}, e(h_1, g_2)^{1/r_2}, X_o^{r_2}) = (c_0^2, c_1^2, c_2^2, c_3^2)$, and generate encrypted keyword information $\mathrm{Tr}(w_1) \leftarrow H_4(e(H_3(w_1), g)^{wk})$. We also generate index blocks $b_1 = (\mathrm{id}_1, (\mathrm{ind} \| \mathrm{op} \| \perp \| \perp) \oplus \mathrm{mask}_1)$ and $b_2 = (\mathrm{id}_2, (\mathrm{ind}_2 \| op \| \mathrm{id}_1 \| \mathrm{key}_1)) \oplus \mathrm{mask}_2$ and store the client status $\sum(w) = c = 1$ (initialization value is -1) at this moment and encrypted keyword information list $\mathbf{T}[\mathrm{Tr}(w_1)] = (\mathrm{id}_2, \mathrm{key}_2)$.

Secondly, we search for the file containing the keyword w_1. The query token of w_1 is generated $\mathrm{Tra}(w_1) = H_3(w_1)^{qk_u}$, the encrypted keyword information $\mathrm{Tr}(w_1)$ is generated from the query token $\mathrm{Tr}(w_1) \leftarrow H_4(e(\mathrm{Tra}(w_1), qk_c))$, and then the search token token = $(\mathrm{id}_2, \mathrm{key}_2)$ is obtained from the $\mathrm{Tr}(w_1)$ query table **T**. Through token token, we find block b_2 in the index. Then, we generate the decryption secret key $\mathrm{mask}_2 \leftarrow H(\mathrm{key}_2, \mathrm{id}_2)$ of b_2 and do XOR operation to get the value of b_2, where $b_2.\mathrm{value} \leftarrow T[\mathrm{id}].\mathrm{data} \oplus \mathrm{mask}$, which contains the information of ind_2 and the information of the previous index block $(\mathrm{id}_1, \mathrm{key}_1)$. Similarly, we get b_1 and decrypt it to get ind_1 and previous index block information $(\perp, \perp)$ and then terminate the search. Through $\mathrm{ind}_{1,2}$, we can find the corresponding ciphertext $C_{1,2}$. In this scheme, the ES is required to reencrypt the ciphertext, and the DU can decrypt it. Taking C_1 as an example, we first generate the reencryption key $rk_{o\longrightarrow u} = (h_2 Y_u Z)^{1/x_o}$, then use it to encrypt the ciphertext, and get the reencrypted ciphertext $C_1' = (c_0^1\prime', c_1^1\prime', c_2^1\prime') = (c_0^1, c_1^1, e(c_3^1, rk_{o\longrightarrow u})/c_2^{1z})$, where $c_2^1\prime' = e(c_3^1, rk_{o\longrightarrow u})/c_2^{1z} = e((h_1^{x_o \cdot r_1}, (h_2 \quad g_1^{y_u} g_2^z)^{1/x_o})/e(h_1, g_2)^{1/r_1 \cdot z}) = e(h_1, h_2 g_1)^{r_1 y_u}$, and the final ciphertext form is $C_1' = (e(h_1, h_2)^{r_1} \cdot m_1, g_1^{r_1}, e(h_1, h_2 g_1)^{r_1 y_u})$.

Finally, we decrypt the final reencrypted ciphertext: $c_0^1\prime' \cdot e(h_1, c_1^1\prime')^{y_u} / c_2^1\prime' = m_1$.

To sum up, the scheme is correct.

6.2. Confidentiality. The EMFSSE scheme is proved to be secure according to the following theorem.

Theorem 1. *In the process of reencryption, the EMFSSE scheme is CPA secure based on the DBDH assumption. Specifically, let A be the adversary of attacking EMFSSE*

scheme with ε advantage in CPA game, and then there is an adversary B who at least solves DBDH assumption with ε/q_{okg} advantage.

Proof. B randomly selects five tuples $T = (g, A = g^a, B = g^b, C = g^c, Z)$ as input. The goal of B is to determine whether Z is random value $e(g,g)^z$ or $e(g,g)^{abc}$ by playing the following game with adversary A, where $a, b, c, z \leftarrow_R Z_p$.

(1) Setup(λ): randomly select $\delta \leftarrow_R Z_p$, set $g_1 = g, g_2 = g^\delta, h_1 = g^a, h_2 = g^b, L = e(g^a, g^b)$, and return $\mathbf{P} = (g_1, g_2, h_1, h_2, L)$. Run $ES - KeyGen(\mathbf{P})$: randomly select $e \leftarrow_R Z_p$, set $sk_e = e$ and $pk_e = g_2^e$, and return pk_e.

(2) Phase 1: A first performs polynomial private times key extraction query. When receiving the query from A, B replies as follows:

(a) $O_{\text{ekg}}(\lambda)$: return the private key of ES $sk_e = e$.
(b) $O_{\text{okg}}(o)$: select $x_o, y_o \leftarrow_R Z_p$ randomly.
(i) If B guesses that it is the target user for this guess, that is, $o' = o$, then calculate $pk_{o'} = (h_1^{x_{o'}}, h_2^{-1} g^{y_{o'}}) = (h_1^{x_{o'}}, g_1^{y_{o'}-b})$ and $sk_{o'} = (x_{o'}, y_{o'} - b)$, hide information $sk_{o'}$, and return $pk_{o'}$.
(ii) Otherwise, $o' \neq o$, calculate $pk_o = (h_1^{x_o}, g_1^{y_o})$ and $sk_o = (x_o, y_o)$, hide information sk_o, and return pk_o.
$O_{\text{ukg}}(u)$: randomly select $x_u, y_u \leftarrow_R Z_p$, let $sk_u = (x_u, y_u)$ and $pk_u = (h_1^{x_u}, g_1^{y_u})$, and return (sk_u, pk_u).
$O_{\text{rkg}}(pk_o, pk_u, pk_e)$, O_{okg} and O_{ukg} generate pk_o and pk_u, respectively; B runs $\text{Reencryption} - \text{KeyGen}(sk_o, pk_u, pk_e)$ and returns reencryption key $rk_{o \longrightarrow u} = (h_2 g_1^{y_u} g_2^e)^{1/x_o}$.

(3) Challenge: when A decides to complete Phase 1, it constructs two equal-length messages $m_{0,1}$ and public key $pk_{o'}$ and sends them to B. B performs the following operations:

(i) If $pk_o \neq pk_{o'}$, B outputs random string and terminates.
(ii) Otherwise, B randomly selects a value $\beta \in \{0,1\}$ and constructs a ciphertext $C = (c_0', c_1', c_2')$ for the challenge, where $c_0' = Z \cdot m_\beta$, $c_1' = g_1^r = g^c$, and $c_2' = e(c_3, W)/c_2^e = e(g,g)^{a(b+1)c^2 \cdot y^u}$, and hides information $r = c$.
Supposing that A does O_{okg} queries at most q_{okg} times, then in Challenge, the probability that B guesses the right o' is at least $1/q_{\text{okg}}$.

(4) Phase 2. B answers A's question, just like in Phase 1. A outputs its guess $\beta' \in \{0,1\}$. If $\beta = \beta'$, B outputs 1; otherwise, it outputs 0; then the advantage of B in solving the DBDH problem is at least $\varepsilon/q_{\text{okg}}$. The proof is complete. □

6.3. Forward Privacy. We refer to the form in the scheme [17] to prove the forward privacy of our scheme. The following is a simplified analysis:

Define $\text{Hist}(w)$ as the history of the keyword w which lists all the modifications to the $\text{DB}(w)$ over a period of time. F is the pseudorandom function controlled by the key, inputs λ bit string and key fk, and outputs λ bit string. $H_{0,1,1',2}$ are secure hash functions defined on random oracle model.

Theorem 2. *L – adaptively – secure of EMFSSE scheme. The leakage function $L_\sum = (L\sum^{Srch}, L\sum^{Updt})$ is defined. If $L\sum^{Srch}(w) = (sp(w), Hist(w))$ and $L\sum^{Updt}(w) = (w, ind) = \perp$, EMFSSE is an $L_\sum$-adaptive security scheme with forward security.*

Proof. Set security parameters λ and μ. Here, we use indistinguishable games derived from the real world to prove the theorem.

(1) Game G_0: G_0 is an SSE security game $\text{SSEReal}_A^{\text{ERMF}}(\lambda, \mu)$ in the real world. The relationship between the two can be expressed as $P[\text{SSEReal}_A^{\text{ERMF}}(\lambda, \mu) = 1] = P[G_0 = 1]$.

(2) Game G_1: G_1 uses random selection instead of calling F to generate keys K_w and K_w^* which are used for generating id and key in ReGen. When a new index (w, ind) is added, the keys K_w and K_w^* are randomly selected and added to the key table KEY. The next time someone asks for the keyword w, it checks the table KEY first. If there is a corresponding key in the table, the stored key is called directly; if not, randomly select the keys K_w and K_w^* and store them in the table. Thus, a reduction is established to distinguish F from real random functions. That is, there are effective adversaries A, $P[G_0 = 1] - P[G_1 = 1] \leq \mathbf{A}\ \mathbf{dv}_{F,A}^{\text{prf}}(\lambda, \mu)$.

(3) Game G_2: G_2 and G_1 have consistent behavior. Except for some inconsistent results of random oracle requests in G_2, the inconsistent time is recorded as Bad. Because the random string u of H_0 is generated in the update operation and updated in the search protocol, there is a time difference Bad. If the adversary asks H_0 before searching, H_0 randomly selects u' to return to the adversary according to the properties of random oracle. After this request, u is programmed to H_0.

(a) Through the consistent transition time node Bad, we can limit the distinguishing advantage between G_2 and G_1: $P[G_2 = 1] - P[G_1 = 1] \leq P[\text{Bad}]$. Probability of Bad: if and only if the adversary queries H_0 in the random oracle model with ind, because the output mask is randomly selected from $\{0,1\}^{\lambda+\mu+1+\log N}$, the probability of

the adversary selecting the corresponding mask is $(1/2^{\lambda+\mu+1+\log N})$. Suppose that a probabilistic polynomial time adversary can query with the random oracle for ploy(λ) times at most; then we can know that $P[\text{Bad}] \leq (\text{ploy}(\lambda)/2^{\lambda+\mu+1+\log N})$ is negligible; that is, $P[G_2 = 1] = P[G_1 = 1]$.

(4) Game G_3: G_3 is similar to G_2 in that it performs the same operation on H_0 in G_2 for $H_{1,1',2}$. The difference is the probability of Bad. If and only if the adversary can query $H_{1,1',2}$ in a random oracle model with correct input, the generated string length is μ, μ, λ, respectively. So the probability of the adversary choosing the right output value is $1/2^{\mu}$, $1/2^{\mu}$, and $1/2^{\lambda}$, respectively. Assuming that a probabilistic polynomial time adversary can perform ploy(γ) queries at most with a random oracle, then $P[\text{Bad}_1] \leq (\text{ploy}(\gamma)/2^{\mu})$, $P[\text{Bad}_2] \leq (\text{ploy}(\gamma)/2^{\mu})$ and $P[\text{Bad}_3] \leq (\text{ploy}(\gamma)/2^{\lambda})$ are all negligible; that is, $P[G_3 = 1] = P[G_2 = 1]$.

(5) Game G_4: G_4 is similar to G_3, but it removes information unrelated to hash functions $H_{0,1,1',2}$; that is, it is a single round trip protocol. Therefore, in the random oracle model, G_4 and G_3 cannot be distinguished; that is, $P[G_4 = 1] = P[G_3 = 1]$.

(6) Simulator S: The code in G_4 is divided into two parts: leak function and simulator. In G_4, the status$u$ of the block is stored, which means that when the time is u, the values in tables $B\text{mask}^*$, $B\text{key}^*$, and $B\text{id}^*$ are mask, key, and id, respectively, and their essence is random string. The simulator uses the counter κ instead of the keyword w. Its initial value is 0, and the value is increased by 1 every time the update operation is performed. The new value is stored in tables $B\text{mask}^*[\kappa]$, $B\text{key}^*[\kappa]$, and $B\text{id}^*[\kappa]$. In the view of adversary A, the update protocol only outputs three random strings, while the search protocol outputs the same number of random strings (key, id, and mask, resp.). So we get the equation $P[G_4 = 1] - P[\text{SSEIdeal}^{\text{ERMF}}_{A,S,L_{\sum}} = 1] = 0$.

Conclusion: According to the above games and simulator S and hash functions $H_{0,1,1',2}$, the following formula can be obtained: $P[\text{SSEReal}^{\text{ERMF}}_{A}(\lambda, \nu) = 1] - P[\ \text{SSEIdeal}^{\text{ERMF}}_{A,S,L_{\sum}} = 1] \leq \mathbf{A}\ \mathbf{dv}^{\text{prf}}_{F,A}(\lambda, \mu)$. □

7. Comparisons and Evaluation

In this section, the proposed scheme is compared with other schemes. And then, we implement the EMFSSE scheme and evaluate its performance.

7.1. Comparisons. Table 3 shows the differences in theoretical computation cost and other functions between EMRF and other schemes (Bost [12] scheme, Chen et al. [21] scheme, and Wang et al. [15] scheme). N is the number of indexes containing the keyword w; E_1 and E_2 represent the multiplication in groups G_1 and G_2, respectively; e is a bilinear pair; M represents secure map-to-point operation; H means the secure hash function; T/T^{-1} represents trapdoor function and its inverse operation (public key primitives, such as RSA), respectively; F/F^{-1} represents a pair of pseudorandom permutations (such as AES or DES).

As shown in Table 3, compared with other relevant schemes, the proposed scheme has better performance. For all schemes, in the update phase, the computational complexity is linearly related to the number of indexes N. Among them, the Bost scheme of [12] has a major flaw that it does public key operation T for N times. Because of its slow encryption, the scheme is limited in large data sets. The computational complexity of Wang's scheme is $2E_1 + e + H + N \times E_2$. Among them, operation e takes a lot of time, but it is independent of index. This scheme calls E_2 once for each index, which leads to relatively high consumption. The computational complexity of Chen's scheme and our scheme is $2E_1 + e + M + F + 6H + N \times (2H)$ and $e + E_2 + T + 7H + 2F + N \times H$, respectively. The operation related to the index number is only the hash function, so it is more efficient than the previous two schemes. However, it is worth noting that Chen's scheme uses map-to-point operation with high cost. To sum up, among the four schemes, our scheme has the best update efficiency.

In the trapdoor generation phase, the complexity of each trapdoor generated by Bost's scheme, Chen's scheme, Wang's scheme, and EMFSSE scheme is H, $e + M + E_1 + H$, $E_1 + H$, and $E_1 + H$, respectively. Obviously, Chen's scheme has a relatively high complexity of generating a trapdoor, while Bost's scheme has relatively optimal complexity. Wang's scheme and our scheme have the same complexity, which is slightly higher than Bost's scheme, but it is a must to extend functionality and improve security. In most cases, the diminished performance is acceptable.

Similar to the update phase, the computational complexity of all schemes in the search phase is also linearly related to N. In order to get each search result, Wang's scheme requires a time-consuming bilinear mapping, and Bost's scheme requires RSA decryption T^{-1}, both of them are expensive. Chen's scheme and EMFSSE scheme only need to perform the corresponding hash function operation, which is less time-consuming. Obviously, the consumption of the EMFSSE scheme is lower.

In terms of function, compared with other schemes, the EMFSSE scheme supports forward privacy and multiuser searchable encryption, which can resist file injection attacks and extend the performance of the scheme to multiuser. In conclusion, the EMFSSE scheme has relatively good performance and low computational costs.

7.2. Evaluation. We do the experiment on the personal computer with Intel Core i5-3337 CPU and 8 GB DDR3 RAM. We use C++ to realize the core function of EMFSSE.

Specifically, we set the security parameters of $\lambda = 128$ bits and $\mu = 64$ bits, use HMAC to implement the hash function, and choose pseudorandom permutation (CBC model, 128-bit key). And we use the 128-bit security level MIRACL CryptoSDK in [36] to calculate the efficiency of pairings and

Table 3: Comparison with other schemes.

Scheme	Update	Trapdoor	Search	Forward privacy	Multiuser
Bost [12]	$H + N \times (2h + T)$	H	$N \times (T^{-1} + 2H)$	√	×
Chen et al. [21]	$2E_1 + e + M + F + 6H + N \times (2H)$	$e + M + E_1 + H$	$4H + N \times (2H)$	√	×
Wang et al. [15]	$2E_1 + e + H + N \times E_2$	$E_1 + H$	$N \times e$	×	×
EMFSSE	$e + E_2 + T + 7H + 2F + N \times H$	$E_1 + H$	$N \times H$	√	√

Table 4: Time consumption of main operations.

Operation	e	E_1	E_2	M	H	F/F^{-1}	T	T^{-1}
Time cost (ms)	105.77	34.45	8.87	322.4	0.001	0.005	0.189	5.896

exponentials operation of bilinear pairing. The time cost of the main operation is shown in Table 4. It takes 105.77 ms to compute bilinear pairs, 34.45 ms to compute exponents in G_1, and 8.87 ms to compute exponents in G_2. It takes 322.4 ms to perform map-to-point operations, and the hash operation takes 0.001 ms. AES algorithm takes 0.005 ms for both encryption and decryption. The execution time of the RSA encryption/decryption algorithm took 0.189 ms and 5.896 ms, respectively.

Figures 3–5 show the approximate time costs of update algorithm, search algorithm, and token generation algorithm with different index numbers ($1.5 \times 10^4 \sim 5.0 \times 10^4$).

Update evaluation: Figure 3 shows the time-consuming comparison of updating algorithms of Bost's scheme, Chen's scheme, and EMRF scheme. (Wang's scheme is not shown in the figure due to its high computational complexity.) As shown in the figure, the abscissa is datasets containing $1 \times 10^4 \sim 5 \times 10^4$ indexes, respectively, with an interval of 0.5×10^4. When the number of indexes is 1×10^4, the cost of Bost's scheme, Chen's scheme, and EMFSSE scheme is 1.91 s, 0.52 s, and 0.12 s, respectively. Bost's scheme has a higher time consumption, Chen's scheme and EPMF have relatively lower time consumption, and EMFSSE has the lowest. The time-consuming of all three schemes increases as the number of indexes increases. Among them, the growth rate of Bost's scheme is the fastest, while that of Chen and EPMF schemes is low and basically flat. In general, the EMFSSE scheme has a lower time-consuming and growth trend during the update phase.

Search evaluation: Figure 4 shows the time-consuming comparison of the search algorithms of Chen's scheme and EMFSSE scheme. (Due to the large time consumption of the other two schemes, no comparison is made.) The abscissa is datasets containing $1 \times 10^4 \sim 5 \times 10^4$ indexes, respectively, with an interval of 0.5×10^4. When the number of indexes is 1×10^4, the cost of Chen's scheme and EMFSSE scheme is 20 ms and 10 ms, respectively. Similar to the update algorithm, the time consumption of the search algorithm also increases with the number of indexes, and the EMFSSE scheme is better than Chen's scheme in both time-consuming and growth trends.

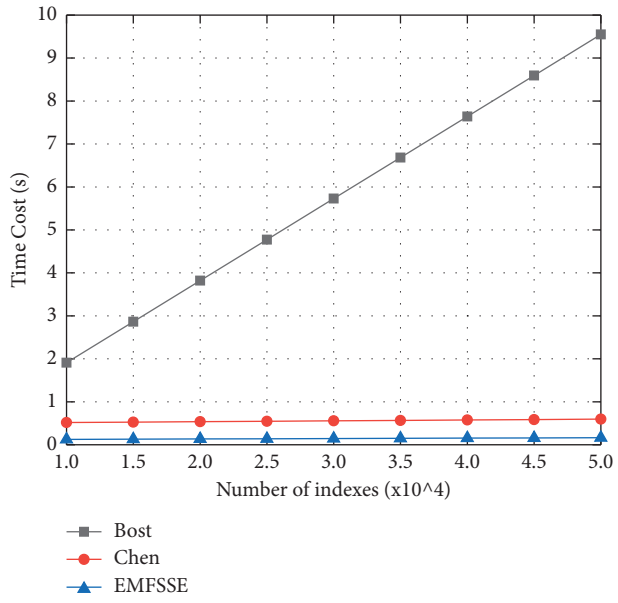

Figure 3: Time consumption of update algorithm.

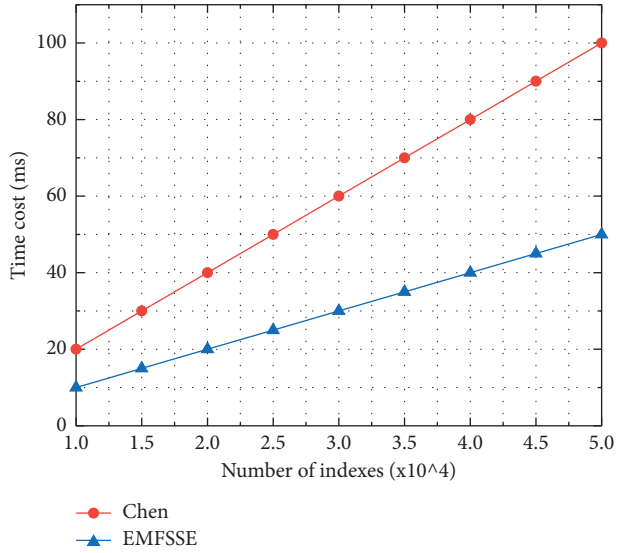

Figure 4: Time consumption of search algorithm.

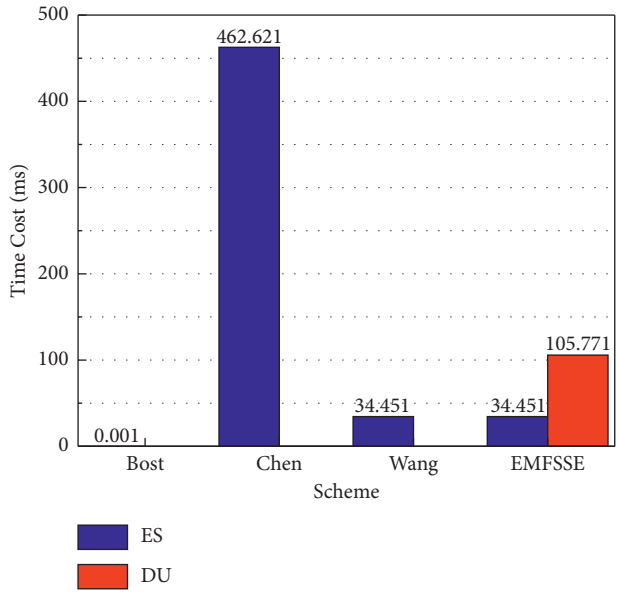

Figure 5: Time consumption of token generation algorithm.

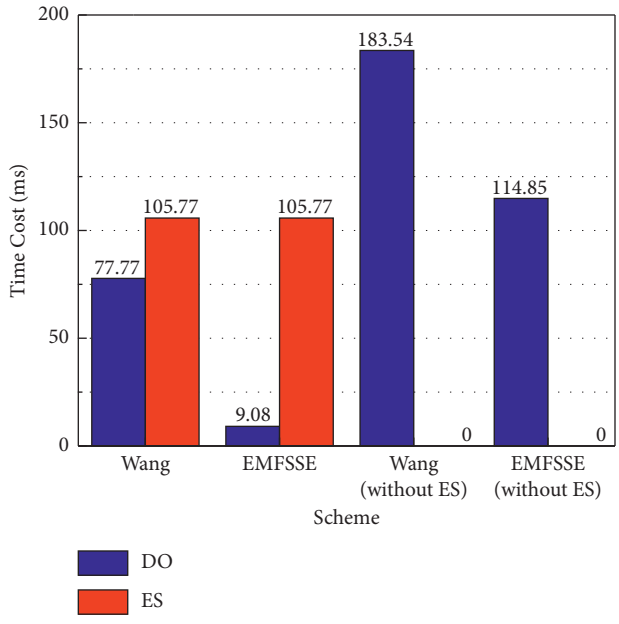

Figure 6: Time cost to generate one ciphertext.

Token generation evaluation: Figure 5 shows the time-consuming comparison of Bost's scheme, Chen's scheme, Wang's scheme, and EMFSSE scheme in a single trapdoor generation algorithm. Chen's scheme needs to do a map-to-point operation, so it is high time-consuming and inefficient. EMFSSE scheme consumes 140.222 ms to generate one trapdoor, but it is worth noting that EMFSSE transfers part of the operation of generating trapdoor to the edge platform, which saves about 75% of the computational costs of the user client. That is to say, the time consumed to generate trapdoor at the DU side is 34.451 ms, which is the same as Wang's scheme. Moreover, compared with the EMFSSE scheme, Wang's scheme can neither support data transmission between multiple users nor meet forward security. In addition, the time EMFSSE scheme consumes is significantly longer than that of Bost's scheme, which only uses the hash function once to generate a trapdoor. But we think that it is necessary to construct a multiuser searchable scheme.

In order to show the advantages of the EMFSSE scheme obtained by using edge computing, EMFSSE is compared with Wang's scheme that also uses edge computing, and the proposed scheme that does not use edge computing. Figure 6 shows the comparison of various schemes for generating a ciphertext. Firstly, the overall time of generating one ciphertext of the EMFSSE scheme is quicker than that of Wang's scheme. Secondly, after using edge computing, the client efficiency of Wang's scheme and the EMFSSE scheme are significantly improved compared to their corresponding schemes without using edge computing. Wang's scheme saves about 57.6% of the client's computing cost, and EMFSSE saves about 92% of the computing cost. As far as we know, in the real environment, edge computing will have better computing efficiency than personal computers, so in the real situation, the computing time of the ES side will be less than our experimental results.

The results show that the EMFSSE scheme has good computational efficiency and security.

8. Conclusion

In this paper, we construct a forward secure searchable encryption scheme supporting multiuser in the P2P network environment. We extend the traditional single-user scheme to multiuser to meet the application requirements of the P2P network environment. And the proxy reencryption technique is used to improve the multiuser scheme. Through reencrypting the searched ciphertext by the proxy party (edge platform), the data user can decrypt the ciphertext with his own private key to avoid direct transmission of the decryption key. In addition, this scheme also uses edge-cloud architecture to replace only-cloud computing and transfers part of the computing to the edge platform, realizing the lightweight computing of this scheme. In terms of security, this scheme satisfies CPA security based on the DBDH assumption in the ciphertext generation process and forward privacy in the index generation process. Theoretical and practical performance analyses show that, compared with the related schemes, this scheme has relatively better efficiency in update operation, search operation, and trapdoor generation operation, and our scheme rarely implements forward security that supports multiuser. In addition, this scheme saves the user's consumption in different degrees in the encryption process or in the token generation process. In the future, we will carry out researches on how to optimize the performance of multiuser searchable encryption scheme by using an attribute-based encryption method.

Data Availability

The data used to support the findings of this study are included within this article.

Conflicts of Interest

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

Acknowledgments

This work was supported by the Research Foundation of Young Core Instructor in Henan Province under Grant 2018GGJS058.

References

[1] N. N. Alleema and D. S. Kumar, "Volunteer nodes of ant colony optimization routing for minimizing delay in peer to peer MANETs," *Peer-to-Peer Networking and Applications*, vol. 13, no. 2, pp. 590–600, 2020.

[2] X. Meng and Y. Deng, "A time-aware resource search strategy with the ant colony optimization in MANETs," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1013–1027, 2019.

[3] S. Zhou, T. Zhang, and X. Meng, "iForest: an informed resource search strategy in mobile P2P networks," *Peer-to-Peer Networking and Applications*, vol. 14, no. 5, pp. 1889–1904, 2021.

[4] X. Li, M. Zhao, M. Zeng et al., "Hardware impaired ambient backscatter NOMA systems: reliability and security," *IEEE Transactions on Communications*, vol. 69, no. 4, pp. 2723–2736, 2021.

[5] X. Li, Q. Wang, M. Liu et al., "Cooperative wireless-powered NOMA relaying for B5G IoT networks with hardware impairments and channel estimation errors," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5453–5467, 2021.

[6] X. Wang, L. T. Yang, D. Meng, M. Dong, K. Ota, and H. Wang, "Multi-UAV cooperative localization for marine targets based on weighted subspace fitting in SAGIN environment," *IEEE Internet of Things Journal*, p. 1, 2021.

[7] G. D. Gonalves, I. Drago, A. B. Vieira, A. P. Couto da Silva, and J. Marques de Almeida, "A study of costs and benefits of content sharing in personal cloud storage," *Journal of Network and Systems Management*, vol. 29, no. 3, 2021.

[8] H. Wang, L. Xu, Z. Yan, and T. A. Gulliver, "Low-complexity MIMO-FBMC sparse channel parameter estimation for industrial big data communications," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 3422–3430, 2021.

[9] H. Teruo, Y. Hirozumi, H. Akihito, U. Akira, and Y. Keiichi, "Edge computing and IoT based research for building safe smart cities resistant to disasters," in *Proceedings of the IEEE 37th International Conference on Distributed Computing Systems*, Atlanta, GA, USA, June 2017.

[10] J. Zhong and X. Xiong, "Data security storage method for power distribution Internet of things in cyber-physical energy systems," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, 15 pages, Article ID 6694729, 2021.

[11] H. Zhong, Y. Zhou, Q. Zhang, Y. Xu, and J. Cui, "An efficient and outsourcing-supported attribute-based access control scheme for edge-enabled smart healthcare," *Future Generation Computer Systems*, vol. 115, pp. 486–496, 2021.

[12] R. Bost, "∑οφος: forward secure searchable encryption," in *Proceedings of the ACM Sigsac Conference on Computer & Communications Security*, ACM, Vienna, Austria, October 2016.

[13] Q. Wang, G. Yu, H. Huang, and X. Jia, *Multi-User Forward Secure Dynamic Searchable Symmetric Encryption*, Springer, Cham, Switzerland, 2018.

[14] B. Lu, J. Zhou, and Z. Cao, "A multi-user forward secure dynamic symmetric searchable encryption with enhanced security," *Journal of Computer Research and Development*, vol. 57, no. 10, pp. 2104–2116, 2020.

[15] W. Wang, P. Xu, D. Liu et al., "Lightweighted secure searching over public-key ciphertexts for edge-cloud assisted industrial IoT devices," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4221–4230, 2020.

[16] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, June 2005.

[17] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proceedings of the network and distributed system security symposium, NDSS*, San Diego, CA, USA, February 2014.

[18] Y. Wei, S. Lv, X. Guo, Z. Liu, Y. Huang, and B. Li, "FSSE: forward secure searchable encryption with keyed-block chains," *Information Sciences*, vol. 500, pp. 113–126, 2019.

[19] J. Li, Y. Huang, Y. Wei et al., "Searchable symmetric encryption with forward search privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 460–474, 2021.

[20] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized I/O efficiency," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 912–927, 2020.

[21] B. Chen, L. Wu, H. Wang, L. Zhou, and D. He, "A blockchain-based searchable public-key encryption with forward and backward privacy for cloud-assisted vehicular social networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5813–5825, 2020.

[22] H. Wang, K. Fan, H. Li, and Y. Yang, "A dynamic and verifiable multi-keyword ranked search scheme in the P2P networking environment," *Peer-to-Peer Networking and Applications*, vol. 13, no. 16, pp. 2342–2355, 2020.

[23] J. Ren, Y. He, G. Yu, and G. Y. Li, "Joint communication and computation resource allocation for cloud-edge collaborative system," in *Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2019.

[24] H. Zhang, S. Chen, P. Zou, G. Xiong, H. Zhao, and Y. Zhang, "Research and application of industrial equipment management service system based on cloud-edge collaboration," in *Proceedings of the 2019 Chinese automation congress (CAC)*, November 2019.

[25] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Secure data sharing and searching at the edge of cloud-assisted Internet of things," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 34–42, 2017.

[26] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," *Advances in Cryptology — EUROCRYPT'98*, vol. 1403Springer, Lecture Notes in Computer Science, , pp. 127–144, 1998.

[27] L. Xu, X. Wu, and X. Zhang, "A certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ACM, Seoul, South Korea, May 2012.

[28] O. Blazy, X. Bultel, and P. Lafourcade, "Two secure anonymous proxy based data storages," in *Proceedings of the SECRYPT*, Lieusaint, France, July 2016.

[29] C. Zuo, J. Shao, J. K. Liu, G. Wei, and Y. Ling, "Fine-grained two-factor protection mechanism for data sharing in cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 1, pp. 186–196, 2018.

[30] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *Proceedings of the Network and Distributed System Security Symposium,NDSS*, San Diego, CA, USA, February 2005.

[31] H. Guo, Z. Zhang, and J. Xu, "Non-transferable proxy re-encryption," *The Computer Journal*, vol. 62, no. 4, pp. 490–506, 2019.

[32] H. Guo, Z. Zhang, J. Xu, N. An, and X. Lan, "Accountable proxy re-encryption for secure data sharing," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 145–159, 2021.

[33] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham, "Sequential aggregate signatures from trapdoor permutations," in *Proceedings of the Advances in Cryptology-EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, May 2-6, 2004.

[34] D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," in *Advances in Cryptology EUROCRYPT 2004*Springer, NY, USA, 2004.

[35] B. Waters, "Efficient identity-based encryption without random oracles," *Advances in Cryptology – EUROCRYPT 2005*, vol. 3494Springer, Lecture Notes in Computer Science, , pp. 114–127.

[36] J. Zhang, Z. Zhang, and Y. Chen, "PRE: stronger security notions and efficient construction with non-interactive opening," *Theoretical Computer Science*, vol. 542, pp. 1–16, 2014.