IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 23, 2020, accepted June 9, 2020, date of publication June 15, 2020, date of current version June 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3002264

Blockchain-Based Searchable Encryption Scheme With Fair Payment

XIXI YAN, XIAOHAN YUAN, QING YE, AND YONGLI TANG
School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454003, China

Corresponding author: Yongli Tang (yltang@hpu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61802117, in part by the Project of Henan Provincial Department of Science and Technology under Grant 192102210280, in part by the Research Foundation of Young Core Instructor in Henan Province under Grant 2018GGJS058, and in part by the Innovative Scientists and Technicians team of Henan Provincial High Education under Grant 20IRTSTHN013.

ABSTRACT In order to solve the problem that most of the existing blockchain-based searchable encryption schemes only support exact keyword search and mutual distrust between the cloud server and the user, a searchable encryption scheme based on blockchain, that has functions such as file update dynamically, search results verification, fuzzy keyword search, and fair payment, is proposed. We use edit distance to generate the fuzzy keywords set. According to the un-modification characteristic of Ethereum blockchain and the security of the RSA accumulator, our scheme can verify the search results by the smart contract and realize the service-payment fairness between the user and the cloud server. The security analysis shows that the proposed scheme achieves non-adaptive semantic security. Performance analysis and experiment show that our scheme is efficient and meets the demand of search application in the cloud environment.

INDEX TERMS Ethereum blockchain, fuzzy keyword search, fair payment, searchable encryption.

I. INTRODUCTION

As an emerging storage model, cloud storage has the feature of pay-per-use, so the cloud server provides its service to the user who is pleasure to pay a certain service fee. Because of its fast and flexible features, more and more users tend to store their data on the cloud server in order to reduce the overhead of local storage space and management. However, as an independent entity, the cloud server may be attacked. Therefore, once the data is stored on the cloud, it will face the risk of being unauthorized accessed, leaked, or modified by the malicious cloud server or the adversary. In order to protect the privacy of the data, the user needs to encrypt data before uploading it to the cloud server. This method not only protects the privacy of sensitive data, but also brings about the problem of searching encrypted documents efficiently. Take electronic medical record application as an example, the patient's disease records, test results, allergy histories, and health status are all outsourced to the cloud server in encrypted form to prevent leakage. When the patient goes to a doctor, he grants the search privilege to the doctor with the medical record card which contains the relevant key, so the doctor can search the related medical records of the patient from the cloud server. First, the doctor generates the search token according to the search key granted by the user, and sends it to the server. Then the cloud server sends back the medical records which contain the searched keyword of the attending doctor. This one-to-one search model not only solves the inconvenience problem of patient's local storage of medical records, but also prevents other doctors from obtaining the patient's privacy. Therefore, Searchable Encryption (SE) technology which realizes the search for encrypted data without revealing the privacy of the user data is proposed [1], and it brings great convenience to the search of patient medical records which require privacy protection.

Traditional searchable encryption schemes are only applicable to the system in which the third-party server is honest and curious. After the user pays the service fee, the server will perform the search service honestly and return the search results. In the practical pay-per-use application under the cloud environment, the cloud server wants to get the service fee before returning the search results, and the user wants to pay the service fee after verifying the correctness of the results. However, the cloud server is a semi-honest and curious entity, so there is a case of fraudulent activity that the server may return partial results or incorrect results after

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

receiving the service fee. At the same time, the user may also malicious and refuse to pay the service fee if he claims the search results to be incorrect even though he receives the correct results. This situation leads to the problem of service-payment unfairness and mutual distrust between the user and the server. Therefore, there is a need for a reliable ciphertext retrieval scheme to solve the problem of service-payment unfairness, which not only detects malicious operation effectively, but also supports fair payment mechanism to punish malicious behavior. To solve the problem of service-payment unfairness, it often depends on a trusted third party, such as the bank. When there is a conflict, the bank needs to spend time to solve it. In order to eliminate the third party organization and realize the fair payment in the point-to-point network, blockchain technology has been concerned.

Due to the features of blockchain such as centerless and un-modification, it can be well combined with cloud computing. Smart contract on the blockchain is a set of agreements, which are directly written into the codes and automatically executed. Smart contract allows trusted transactions and protocols to be executed without the involvement of a central authority. Therefore, blockchain and smart contract are suitable for performing verification operation in the searchable encryption system to achieve service-payment fairness. The searchable encryption scheme that blockchain payment fairness mechanism is introduced into encourages the participants to perform the correct operation according to the rules of the contract and realize the service-payment fairness. As long as the user and the cloud server execute honestly according to the rules of the smart contract, the cloud server can obtain the corresponding service fee, and the user gets the correct search results at the same time. Once the cloud server is detected to be dishonest, it will be penalized by losing the guarantee deposit instead of getting a service charge. Under the premise that the search results are correct, the user cannot deny the results provided by the cloud server and refuse to pay the service fee. Therefore, searchable encryption based on blockchain ensures service-payment fairness between the user and the cloud server.

There are related research literatures [22]–[33] on the blockchain-based searchable encryption that only support exact keyword search and allow the user to search encrypted data through keyword, but the scheme supports fuzzy keyword search is still lacking. That is, spelling mistake cannot be tolerated. For example, the keyword "keyword" is stored on the server, but the user accidentally types in "kayword" to search, which is typical search behavior and very common in real life. In practical applications, it is inevitable that the keyword is misspelled in user's search request, resulting in a large number of irrelevant documents are returned to the user. In this case, the availability of the system is affected, and the user enjoys a bad search experience. Therefore, when the user inputs a misspelled keyword, the fuzzy keyword searchable encryption algorithm can return the closest possible matching documents. It can be particularly used to make a searchable encryption system tolerant to user's typos. In searching, it is possible that the user enters an incorrect query, but the user still looks for relevant documents to be retrieved. Fuzzy keyword searchable encryption based on blockchain retrieves the documents matching the search query exactly and retrieves the closest possible matching documents if there are minor inconsistencies in the search request. It greatly enhances the usability of the blockchain-based searchable encryption system. In addition, when the data owner outsources a large amount of data to the cloud server, it is necessary to consider the problem of updating of data in the cloud storage, such as addition, deletion, modification, and so on. Therefore, the searchable encryption with fuzzy keyword search that supports the search results verification and data update dynamically is particularly important.

A. OUR CONTRIBUTION

Although existing blockchain-based searchable encryption schemes allow a user to securely search over encrypted data through keyword, these schemes only support exact keyword search. That is, there is no tolerance of minor typos which is a typical search behavior and very common in real life. This weakness makes the usability of existing blockchain-based searchable encryption schemes is greatly affected. Blockchain-based fuzzy keyword searchable encryption can return the closest possible matching documents when the user's inputted keyword is misspelled. It can be used to make a blockchain-based searchable encryption system tolerant to user's typos and greatly enhance the usability of the blockchain-based searchable encryption system. In addition, when the data owner outsources a large amount of data to the cloud server, it is necessary to consider the problem of updating of data in the cloud storage, such as addition, deletion, modification, and so on. Therefore, the searchable encryption with fuzzy keyword search that supports the search results verification and data update dynamically is particularly important. Although previous scholars have studied fuzzy keyword search, dynamic and verifiability, and introduction of blockchain mechanism in searchable encryption schemes, there is still no blockchain-based searchable encryption scheme that supports the fuzzy keyword search, and has the properties of dynamic storage mechanism and verifiability of the results. Therefore, dynamic and verifiable fuzzy keyword searchable encryption based on blockchain is proposed in our paper. The main contributions are as follows:

(1) Our scheme supports dynamic update of the document, and uses smart contract to complete the verification of the search results. At the same time, the blockchain payment fairness mechanism is introduced into the searchable encryption scheme. A verification smart contract designed to implement a fair payment mechanism can automatically detect the malicious operation of the user or the cloud server. According to the un-modifiability of Ethereum blockchain and the security of the RSA accumulator, as long as the user and the cloud server perform honestly according to the contract rules, the

user can obtain correct retrieval results without local additional verification which can reduce the computational overhead of the data user.

(2) Our scheme realizes fuzzy keyword search. Our scheme combines fuzzy keyword search, blockchain technology, and symmetric searchable prime encryption. We construct fuzzy keywords set by using edit distance. The cloud server can directly decrypt the corresponding encrypted index vector by the searched fuzzy keyword without interacting with the user multiple times. When the user inputs a misspelled keyword, our scheme can successfully infer the meaningful keyword from the error keyword, and return the matched documents as close as possible.

(3) The security of our scheme is non-adaptive semantic security. Comparative analysis and experimental results show that the feasibility and validity of our scheme, which meets the search application requirement in the cloud environment and has the functions of file update dynamically, search results verifiability, fuzzy keyword search, and fair payment.

B. RELATED WORK

1) SEARCHABLE ENCRYPTION

In order to realize ciphertext retrieval, Song *et al.* [1] proposed a searchable encryption scheme based on the ciphertext scanning idea. Goh [2] proposed an index-based scheme that Bloom filter was used as an index structure to determine whether a ciphertext file contains a specific keyword. Curtmola *et al.* [3] normalized the definition and security objective of Symmetric Searchable Encryption (SSE) and proposed SSE-1 and SSE-2 schemes that achieved indistinguishable security under non-adaptive and adaptive attack models. Since the documents that stored on the cloud are updated dynamically, combining the SSE-1 scheme [3] with the "file-keyword" index structure, Kamara *et al.* [4] proposed a SSE scheme that supported sublinear time keyword retrieval and efficient dynamic update of ciphertext files. Kamara and Papamanthou [5] introduced the keyword red-black tree as an index structure and enabled dynamic SSE to support the parallel process in multi-processor. Guo *et al.* [6] used the Bloom filter to construct an index tree, and implemented multi-keyword ranked searchable encryption in the case of file update dynamically. In the actual application of the cloud environment, the cloud server is a semi-honest and curious entity, the user needs to verify the search results. Chai and Gong [7] introduced a searchable hash tree and proposed a verifiable SSE scheme. Kurosawa and Ohtaki [8] introduced the concept of privacy and reliability to propose a universal-composability secure verifiable SSE scheme. Jiang *et al.* [9] constructed a verifiable multi-keyword ranked searchable scheme that the client used the Message Authentication Code (MAC) and binary vector to verify the correctness of the returned results. Wu *et al.* [10] proposed a verifiable public key searchable encryption scheme by using homomorphic encryption in a multi-user environment. Kurosawa and Ohtaki [11] proposed a dynamic and verifiable searchable encryption scheme in which the data user can update files dynamically and detect malicious behavior of the server. Sardar and Ruj [12] used a bilinear digital signature based on a dynamic SSE scheme to achieve forward security and public verifiability. Ramasamy *et al.* [13] used the bitmap index and homomorphic MAC to dynamically update the documents and verify search results without leaking the access pattern. However, most of the existing verifiable searchable encryption schemes only focus on detecting malicious behavior, but lack a mechanism to punish dishonest performers. Although more and more efficient Symmetric Searchable Encryption (SSE) schemes are designed and deployed in practical applications, they are affected by some information leakage. Therefore, scholars try to attack SSE schemes with some degree of information leakage [14]–[21]. In order to make searchable encryption more suitable for the needs of reality, the literatures [53]–[57] have studied various searchable encryption schemes. Li *et al.* [54] presented an attribute-based encryption scheme with outsourcing key issuing and outsourcing decryption, which can implement keyword search function. Li *et al.* [55] came up with a searchable ciphertext policy attribute-based encryption scheme with attribute revocation, where access structures were partially hidden so that receivers cannot extract sensitive information from the ciphertext. Lu *et al.* [56] developed a privacy-preserving and pairing-free multi-recipient certificateless encryption with keyword search scheme for the cloud-assisted industrial internet of things. Lu *et al.* [57] devised a certificate-based searchable encryption scheme, which not only provided resistance to the keyword guessing attack, but also had advantages such as implicit authentication, no key escrow and no secure channel.

2) BLOCKCHAIN-BASED KEYWORD SEARCH

In order to achieve a fair search on the blockchain, Cai *et al.* [22] implemented dynamic and reliable keyword retrieval in the distributed storage, and stored the metadata of the search results as unforgeable evidence on the blockchain for a fair search between the client and the server. Li *et al.* [23] adopted Bitcoin time-lock technology to implement search-payment fairness, stored encrypted data and secure index on the blockchain, and proposed two different single-keyword searchable encryption schemes according to the size of the data. Cai *et al.* [24] implemented a trustworthy private keyword search in a distributed storage system, and solved the fairness problem by combining blockchain technology with an efficient dynamic searchable encryption scheme. Do and Ng [25] built a secure distributed storage and private keyword searchable system and realized the integrity authentication of retrieval results with the help of blockchain by storing encrypted data on distributed nodes and storing data fingerprint on the blockchain. Cai *et al.* [26] proposed a distributed storage framework which supported private keyword search. In this framework, the fingerprint of the encrypted documents, the search token, and the metadata for integrity verification were stored on the blockchain, so that the client authentication and fair payment could be obtained. Hu *et al.* [27] used a designated smart contract instead of a central server to build a distributed privacy-protection searchable encryption scheme, in which only by enforcing the contract rules honestly can participants get the right search results and service fee. Zhang *et al.* [28] proposed a blockchain-based outsourcing service fair payment framework named BCPay in the cloud computing environment, and applied the BCPay in a searchable encryption scheme, which used the Bitcoin-timing commitment scheme to realize the fairness of the search service and ensure that the honest participants who were in legal behavior could obtain search results and service fee without needing a trusted third party. Wang *et al.* [29] proposed a distributed storage system with fine-grained access control based on blockchain. By removing the cloud server and using a smart contract to store secure index and perform the search, their scheme solved the problem of incorrect results returned by the cloud server and realized the fair keyword retrieval function on the encrypted data. Li *et al.* [30] adopted time-lock technology and bitcoin blockchain to realize a single-keyword SSE scheme, which solved the fairness problem of the SSE scheme without removing the cloud server and verified the correctness of the search results through the script of bitcoin. Zhang *et al.* [31] built a secure index based on the digital signature, allowed the user to implement a trusted keyword search on encrypted data and verify the correctness of the search results. The server-side verifiability was first proposed in their scheme, which means that the cloud server could be protected from being framed by the malicious data owner in the data storage stage. Zhang *et al.* also employed the blockchain technology and hash function to realize fair payment of the search fee without a third party. By storing the verification index on the blockchain and matching the search results of the search index in the cloud server, Wang *et al.* [32] proposed a verifiable searchable encryption scheme for a single keyword search according to the irreversible feature of the blockchain. Chen *et al.* [33] built an index through logical expressions and stored them on the blockchain, so that the user can search expressions. The use of blockchain ensured the correctness of the search results without local verification of the user, and realized the fairness of service-payment. However, most of the existing schemes focus on exact keyword search and require that only the keyword entered by the user must be exactly matched with the predefined keywords could the search results be returned, whereas the fuzzy keyword searchable encryption based on blockchain gets less attention. Therefore, the blockchain-based searchable encryption scheme needs to support fuzzy keyword search, i.e., the user inputs a misspelled keyword, the server should return the matched documents which are as close as possible to the user's interest.

3) FUZZY KEYWORD SEARCH

Exact keyword searchable encryption lacks tolerance for the misspelled keyword. In order to realize the fuzzy keyword search, Li *et al.* [34] proposed a fuzzy keyword retrieval scheme for the first time which used edit distance to define and measure the similarity between the keywords, and constructed two fuzzy keywords sets based on the wildcard and gram methods. Wang *et al.* [35] further studied the fuzzy keyword retrieval scheme and gave a formal security proof. Bösch *et al.* [36] proposed a conjunctive wildcard searchable scheme that inserted keywords into the Bloom filter, and used a pseudo-random number based on the file identifier to confuse the binary vector of the Bloom filter. The scheme generated all keywords with wildcard retrieval form in advance, and then converted fuzzy keyword search into exact keyword search. Ge *et al.* [37] proposed a verifiable fuzzy keyword searchable encryption scheme, but the user and the cloud server needed to interact multiple times during the search phase. Tahir *et al.* [38] implemented search results ranked based on the fuzzy keyword search. Wang *et al.* [39] constructed a verifiable fuzzy keyword searchable scheme, which not only could perform the fuzzy keyword search on encrypted data, but also protected the privacy of keywords and achieved the verifiability of search results. Zhu *et al.* [40] proposed a verifiable and dynamic fuzzy keyword searchable scheme, which had the feature of universal-composability security against malicious attack, and used the RSA accumulator [41] to verify the correctness of the search results. The research of fuzzy keyword search can be referred to the literature [42]–[49]. Security comparison of fuzzy keyword searchable encryption schemes [34]–[40], [42]–[48] as shown in Table. 1. In Table. 1, $\surd$ represents the scheme uses non-adaptive attack model, or the proof of the scheme is simulation-based, or the adversary type of the scheme is honest but curious clod server.

C. ORGANIZATION

The rest of this paper is organized as follows. Section II gives relevant preliminaries. In section III, we explain the system model, threat model, and relevant algorithm definitions. Section IV constructs a dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain. Section V analysis the scheme from the point of security, fairness, function features, and gives experiment results based on the real data set. We give the conclusion and further discussion in section VI.

II. PRELIMINARIES

Before introducing the specific scheme, Table 2 gives the symbols and descriptions used in our scheme.

A. SYMMETRIC SEARCHABLE ENCRYPTION

The data owner encrypts the plaintext documents D to obtain the encrypted documents C and secure index I with the symmetric key K, then uploads them together to the cloud

TABLE 1. **Security comparison of fuzzy keyword searchable encryption schemes.**

Scheme	Security model: Non-adaptive attack model	Security proof: Simulation-based	Honest but curious clod server
[34]	√	√	Semi trusted but curious clod server
[36]	Adaptive attack model	√	√
[37]	√	√	Semi trusted but curious clod server
[38]	√	Indistinguishabilit y-based	√
[39]	√	Indistinguishabilit y-based	Semi trusted but curious clod server
[40]	√	√	Semi trusted but curious clod server
[35, 42, 43, 45, 46]	√	√	√
[44]	Security description	√	√
[47]	√	Security description	√
[48]	Security description	Security description	Semi trusted but curious clod server
Our scheme	√	√	Malicious clod server and DU

TABLE 2. **Symbols and descriptions.**

Symbol	Description
D	Collection of plaintext documents $D=\{D_1,D_2,\cdots,D_n\}$
C	Collection of encrypted documents $C=\{C_1,C_2,\cdots,C_n\}$
D_w	Collection of plaintext documents $D_w=\{D_{w,1},D_{w,2},\cdots,D_{w,j}\}$ containing the keyword w
C_w	Collection of encrypted documents $C_w=\{C_{w,1},C_{w,2},\cdots,C_{w,j}\}$ that contains the keyword w
W_D	Collection of keywords $W_D=\{w_1,w_2,\cdots,w_m\}$ extracted from D
T_w	The search token of keyword w
$v(w)$	The index vector of keyword w
$E(w)$	The encrypted index vector of keyword w
I	Secure index
$S_{w,d}$	The fuzzy keywords set of keyword w with edit distance d
$w_{i,t}$	The keyword $w_{i,t}$ in fuzzy keywords set $S_{w,d}$, where $1\le t\le\|S_{w_{i,d}}\|$
n	The number of documents in the collection D
m	The number of keywords in the collection W_D
d	Edit distance
[n]	The set of integers $\{1,2,\ldots,n\}$, where n is an integer
$acc(C)$	The accumulated value of documents set C
$pf(C)$	The verification evidence of documents set C
\$ Guaranty	Guarantee deposit submitted by the cloud server
\$ Fee	Service fee submitted by the user

server. The data owner shares the key K with the user through a secure channel. When the user searches a keyword w, the corresponding search token T_w is generated by the key K and sent to the cloud server. The cloud server matches the search token T_w with the secure index I to obtain the target encrypted documents C_w which contain the searched keyword w, and returns it to the user. The data user decrypts the target documents C_w to obtain the plaintext documents D_w.

B. EDIT DISTANCE AND FUZZY KEYWORD SEARCH

There are many approximate string-matching algorithms to measure the string similarity. In the fuzzy keyword searchable encryption scheme based on blockchain, the well-studied edit distance from the literature [34] is used to evaluate the similarity between the two strings. For two keywords w_1 and w_2, the edit distance $d(w_1, w_2)$ refers to the minimum number of single-character operations required to convert one of the keywords to another. There are only three single-character executable operations:

Insert: insert a character.

Delete: delete a character.

Substitution: convert a letter to another letter.

For example, a single-character operation from keyword *kitten* to keyword *sitting*:

(1) $kitten \rightarrow sitten(k \rightarrow s)$

(2) $sitten \rightarrow sittin(e \rightarrow i)$

(3) $sittin \rightarrow sitting(insert\ g)$

So, the edit distance from the keyword *kitten* to the keyword *sitting* is $d(kitten, sitting) = 3$.

In order to realize fuzzy keyword searchable encryption based on blockchain, our paper uses edit distance to construct fuzzy keywords set. Given a keyword w and edit distance d, we let $S_{w,d}$ denote fuzzy keywords set that satisfies $S_{w,d} = \{w'|d(w, w') \le d\}$. Data owner forms a secure index by encrypting each keyword in the fuzzy keywords set $S_{w_i,d} (1 \le i \le m)$. After receiving the search token containing the keyword queried by the user, the cloud server matches it with keyword ciphertext in the index one by one. If matching is successful, all relevant documents will be returned to the user. Otherwise, the search fails.

C. ETHEREUM BLOCKCHAIN

Ethereum is a distributed application platform for the smart contract. It extends the function of Bitcoin and supports Turing-complete script language. It is a programmable blockchain system.

In Ethereum, there are two different types of accounts: an externally owned account and a contract account. The externally owned account is controlled by the private key, whereas the address corresponds to the public key. The externally owned account can initiate a message communication transaction for transferring or create a contract creation transaction to trigger the execution of the contract codes. The contract account is controlled by the contract codes. Once created, its codes will be activated and run on the blockchain with no change.

The transactions in Ethereum are divided into communication transaction and contract creation transaction. The communication transaction refers to a transaction that transfers money from one external account to another. The contract creation transaction represents

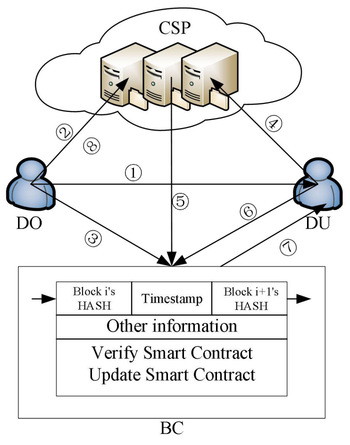

FIGURE 1. System model.

the creation of a new Ethereum smart contract and triggers the execution of the relevant codes in the smart contract.

The Ethereum smart contract is a set of agreements that control the digital assets and contain the rights and obligations of the contract participants, which are automatically executed according to the prior rules by the computer without human involvement. The smart contract is written into the Ethereum blockchain in a digital form. The blockchain will ensure that the smart contract is transparent, traceable, un-modifiable, and undeniable during the entire process of storage, reading, and execution because of the feature of un-modifiable and hash algorithm. In Ethereum, the smart contract is a special account that consists of an account address, script codes, balance, and storage space.

D. RSA ACCUMULATOR

According to literatures [40], [41], we briefly introduce the principle of RSA accumulator. RSA accumulator is an efficient data authentication method. After entering an arbitrarily large set, it outputs a constant size digest and a constant size witness for any element in the set to verify the (non-)membership of the element in this set. Suppose G is a cyclic group and g is a generator of G. Suppose x, y are primes. Let $N = xy$ and $G = \{u = v^2 \mod N | v \in Z_N^*\}$. For a set $S = \{s_1, \cdots, s_n\}$, the calculated value of RSA accumulator is $acc(S) = g^{\prod_{i=1}^{n} P(s_i)} \mod N$, where $P(s_i)$ is a randomly chosen prime generated by s_i. For any element $s_j \in S$ in the set S, a proof $pf(S) = g^{\prod_{i \neq j} P(s_i)} \mod N$ is generated. Finally, we can verify the element s_j is in the set by checking $acc(S) \overset{?}{=} pf(S)^{P(s_j)} \mod N$.

III. PROBLEM FORMULATION

A. SYSTEM MODEL

The system model of dynamic and verifiable fuzzy keyword searchable encryption based on blockchain is shown in Fig. 1. It contains four participants, namely Data Owner (DO), Data User (DU), Cloud Service Provider (CSP), and Ethereum Blockchain (BC).

(1) DO and DU. DO has the responsibility to grant search permission to DU. DU is granted search right by DO, and they share the symmetric key and search key through the secure channel.

(2) Suppose the DO has n plaintext documents $D = \{D_1, D_2, \cdots, D_n\}$ that need to be encrypted and uploaded to the CSP. The DO extracts a collection of keywords $W_D = \{w_1, w_2, \cdots, w_m\}$ from the plaintext documents set D. DO generates a fuzzy keywords set $S_{w_i,d}$ for each precise keyword by using edit distance, where $1 \leq i \leq m$. Based on the fuzzy keywords set, DO builds a secure searchable index I. The DO uses a symmetric encryption algorithm to generate a collection of encrypted documents $C = \{C_1, C_2, \cdots, C_n\}$, such as AES. The DO uploads the secure index I and the encrypted documents collection C to the cloud server together.

(3) To verify the search results, DO generates an accumulated value $acc(C)$ for the encrypted documents set C. Then DO deploys the verification contract and stores the accumulated value $acc(C)$ on the contract.

(4) When the authorized DU wants to obtain the documents containing the searched keyword w', it uses the search key to calculate the search token $T_{w'}$ and sends it to the CSP as a search request. We allow the DU to input a misspelled keyword w' and search for the closest documents in our paper.

(5) After receiving the search token from the DU, the CSP performs search operation on the secure index I and the encrypted documents C to obtain encrypted documents set $C_{w'}$ containing the searched keyword w' and the verification evidence $pf(C)$. Then the CSP stores them on the smart contract along with the deposit Guaranty submitted by the CSP.

(6) DU calls the *verify* function of the verification smart contract and stores the service fee Fee temporarily on the smart contract.

(7) The verification contract verifies the search results that returned by the CSP. When the verification fails, the CSP loses the deposit as a penalty, and the DU redeems the service fee. When the verification passes, the DU obtains the correct search results $C_{w'}$ without local verification, which will reduce the calculation overhead of the DU. At the same time, the CSP redeems the deposit and receives the service fee to achieve service-payment fairness.

(8) When the verification passes, the DU decrypts the ciphertext documents received from the contract through the symmetric key.

(9) In the actual application, DO can add, delete, or modify document dynamically. So it regenerates a new index I', encrypted document C', and accumulated value $acc(C)'$ to implement update dynamically.

B. ALGORITHM DEFINITION

Definition 1: A dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain consists of eight probabilistic polynomial-time algorithms $\prod = (Setup, KeyGen, Enc, TokenGen, Search, Verify, Dec, Update)$:

(1) System initialization algorithm. $Setup(1^{\lambda}) \rightarrow (MK, PK)$. The DO inputs security parameter λ, and outputs the system master private keys MK and public parameters PK.
(2) Key generation algorithm. $KeyGen(1^{\lambda}) \rightarrow (sk, k_1)$. The DO takes a security parameter λ as input, outputs symmetric key sk and search key k_1. They are shared with the authorized DU together through the secure channel.
(3) Encryption algorithm. $Enc(PK, D, sk, k_1) \rightarrow (C, I, acc(C))$. The DO inputs public parameters PK, plaintext documents collection D, sk, and k_1. Then the algorithm outputs encrypted documents set C, secure index I, and accumulated value $acc(C)$. The DO uploads C and I to the CSP together. Then DO deploys the verification contract and stores $acc(C)$ on the contract.
(4) Search token generation algorithm. $TokenGen(PK, w', k_1) \rightarrow T_{w'}$. The DU inputs public parameters PK, the searched fuzzy keyword w', and the shared search key k_1, then the algorithm outputs a search token $T_{w'}$ and sends it to the CSP. Because our scheme implements the fuzzy keyword search, the searched keyword is allowed to be misspelled.
(5) Search algorithm. $Search(PK, C, I, T_{w'}) \rightarrow (C_{w'}, pf(C))$. After the CSP inputs public parameters PK, the encrypted documents set C, secure index I, and the search token $T_{w'}$, it outputs the encrypted documents $C_{w'}$ which the closest to the searched keyword and the evidence $pf(C)$ for correctness verification. The CSP sends them to the smart contract.
(6) Verification algorithm. $Verify(PK, C_{w'}, acc(C), pf(C)) \rightarrow C_{w'}/\perp$. The Ethereum smart contract inputs public parameters PK, search results $C_{w'}$, accumulated value $acc(C)$, and verification evidence $pf(C)$. When the verification passes, it indicates that the CSP provides the correct results, then the contract returns $C_{w'}$ to the DU reducing the computational overhead, and realizes the service-payment fairness, otherwise, it outputs $\perp$.
(7) Decryption algorithm. $Dec(sk, C_{w'}) \rightarrow D_{w'}$. After the DU obtains the correct search results $C_{w'}$, it decrypts search results by using the symmetric key sk and gets plaintext documents $D_{w'}$.
(8) Update algorithm. $Update(PK, acc(C), D_{n+1}, D_j) \rightarrow (C', I', acc(C)')$. The DO can add a document D_{n+1}, delete a document D_j, or modify a document D_j dynamically, and it outputs the updated encrypted document C', secure index I', and accumulated value $acc(C)'$.

C. THREAT MODEL

In the dynamic and verifiable fuzzy keyword searchable encryption based on blockchain, the threat model is that CSP and DU are considered to be "malicious".

In order to save storage space, communication bandwidth, and computing cost, the CSP may delete some encrypted documents or forge search results, and there is curiosity to try to deduce certain keywords and user's privacy information based on the communication on the public channel, such as secure index, encrypted documents, and search token. Besides the encrypted documents, the secure index, and the search trapdoor, the cloud server can also know and record each search result. At the same time, the malicious CSP can return partial results or incorrect results after receiving the service fee to defraud the service charge, which causes DU to lose the fee and get the wrong results, resulting in an unfair situation to DU in the cloud environment of pay-per-use.

In addition, there is a phenomenon that malicious DU obtains the search results fraudulently and refuses to pay the service fee after enjoying search service of the CSP, causing an unfair situation to the CSP.

D. SECURITY MODEL

We refer to the non-adaptive attack model proposed in [3]. The non-adaptive attack model only considers the adversary who cannot select search request based on the search token and the previous search results. This is acceptable because only the user with the search key can generate the search token. Similar to the scheme in [3], our scheme always generates the same search token for the same searched keyword and returns the same results. The CSP cannot know any additional information except access pattern (by observing the encrypted documents in the search results) and search pattern (by observing the search request). Therefore, security should ensure that no information other than the results of a series of search requests, update requests, and access pattern are disclosed. The result of the update request can also be seen as part of the search pattern. Most of the existing searchable encryption schemes cannot protect search pattern and access pattern, we do not aim to protect them in this work. We use the security model defined in [3] by performing a simulation-based game between the adversary and the simulator, which allows leaking the access pattern and the search pattern to prove security.

We introduce some auxiliary concepts used in [3] and apply them to our proposed scheme.

- *History* is $H = (D, Q, D_{n+1}, D_j)$, where D is a collection of documents, $Q = (w_1, w_2, \cdot\cdot\cdot, w_q)$ is a set of searched keywords, D_{n+1}, D_j are the documents used in the update operation.
- *View* is $V(H) = (C, I, T_Q, C', I')$, where C is a collection of encrypted documents (encrypted by using symmetric key sk), I is an encrypted secure index (encrypted by using the key k_1 of pseudo-random function), $T_Q = \{T_{w_i}\}_{i\in[q]}$ is an encrypted search token of searched

keywords (encrypted by using the key k_1), C' is the ciphertext of the documents D_{n+1} and D_j in the update operation, and I' is the updated index. Given history H, the CSP can only see the encrypted information of history H, i.e., view $V(H)$.

- *Trace of history* is $Tr(H) = (n = |D|, \{|D_i|, id(D_i)\}_{i\in[n]}, m = |W_D|, |S_{w_i,d}|_{i\in[m]}, |D_{n+1}|, |D_j|, id(D_{n+1}), id(D_j), AP(W), SP(Q))$, which contains the sensitive information that can be exposed to the CSP, such as access pattern and search pattern. Since ciphertext documents and secure index are stored in the CSP, *Trace of history* contains the number of documents in the documents collection $n = |D|$, the size and identification of each document $\{|D_i|, id(D_i)\}_{i\in[n]}$, the number of keywords in the documents collection $m = |W_D|$, the size of the fuzzy set of each keyword $|S_{w_i,d}|_{i\in[m]}$, and the size and identification of each updated document $|D_{n+1}|, |D_j|, id(D_{n+1}), id(D_j)$. In the process of dynamic update, it is allowed to disclose the size of the updated file, the number of keywords, and the size of the fuzzy keywords set, all of which are included in *Trace of history* of our security definition. After the update is completed, the elements in *Trace of history* which described the leakage of information, will be updated accordingly, such as the number of the documents, the size of keywords, etc.. Thus, the result of the update request can also be seen as part of the information allowed to leak. $AP(W)$ is the access pattern of each keyword, i.e., $(id(D_1), \cdots, id(D_{|D(w_i)|}), |D_1|, \cdots, |D_{|D(w_i)|}|)_{i\in[m]}$. $SP(Q)$ is the search pattern used to describe whether any two search requests contain the same keyword. $SP(Q)$ is a symmetric binary matrix such that for $1 \le i, j \le q$, the element in the i-th row and j-th column is 1 if $w_i = w_j$, otherwise, is 0.

Definition 2 (Non-Adaptive Semantic security): Let π denote a dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain, λ be the security parameter, A be an adversary, S be a simulator. We define the following probability experiments $Real^{\pi}_{A}(\lambda)$ and $Sim^{\pi}_{A,S}(\lambda)$, where $Real^{\pi}_{A}(\lambda)$ is carried out between the challenger C and the adversary A, $Sim^{\pi}_{A,S}(\lambda)$ is performed between the adversary A and the simulator S. When given two histories H, H^* with the same trace, i.e., $Tr(H^*) = Tr(H)$, they generate $V(H)$, $V^*(H^*)$, respectively.

```
$Real^{\pi}_{A}(\lambda)$ :
    $(sk, k_1) \leftarrow KeyGen(1)^{\lambda}$
    $H \leftarrow A$
    Extract $(D, Q)$ from $H$
    $(C, I) \leftarrow Enc_{sk}(D)$
    For $1 \le i \le q$
      $T_{w_i} \leftarrow TokenGen(k_1, w_i)$
    Let $T_Q = (T_{w_1}, \cdots, T_{w_q})$
    $H \leftarrow A$
    Extract $D_{n+1}/D_j$ from $H$
    $(C', I') \leftarrow Update(D_{n+1}, D_j)$
    Update the related elements in $Tr(H) = Tr(H^*)$
    Output $V(H) = (C, I, T_Q, C', I')$
$Sim^{\pi}_{A,S}(\lambda)$ :
    $H^* \leftarrow A$
    $V^*(H^*) \leftarrow Sim(Tr(H^*))$, where $Tr(H^*) = Tr(H)$
    Output $V^*(H^*) = (I^*, C^*, T^*_Q, C'^*, I'^*)$
```

We say that π is semantically secure if for all probabilistic polynomial time adversaries A, there exists a probability polynomial-time simulator S such that

$$|\Pr[V(H) \leftarrow Real^{\pi}_{A}(\lambda)] - \Pr[V^*(H^*) \leftarrow Sim^{\pi}_{A,S}(\lambda)]| \le negl(\lambda),$$

where $negl(\lambda)$ is a negligible function. That is to say, when given two histories H, H^* with the same trace, if $V(H)$ and $V^*(H^*)$ are not distinguishable, the adversary cannot get any additional knowledge except for the access pattern and search pattern. Thus, our scheme is non-adaptive semantic security.

E. DESIGN GOALS

Based on the above threats, the design goals are as follows:

(1) **Fuzzy keyword search.** The SE scheme, which supports fuzzy keyword search, aims to tolerate typos in inputting of the user. We describe the design goal of fuzzy keyword search as follows. Given a plaintext documents set D, a ciphertext documents set C, a keywords set $W_D = \{w_i | i \in [m]\}$ extracted from D, a keyword w' searched by the user, and a specified edit distance d, the fuzzy keyword search is performed as follows: 1) If $w' = w_i \in W_D$, that is, the user's search input exactly matches the pre-set keyword, then the server is expected to return $C_{w_i} = C_{w'}$ containing the keyword. 2) If $w' \notin W_D$, that is, the user's search input is misspelled, then the server will return the closest possible results C_{w_i} based on the pre-specified fuzzy keywords set, where $d(w', w_i) \le d$, i.e., w' is the one of elements in the fuzzy keywords set of w_i. When the user inputs a misspelled keyword, the scheme should return the documents that match the user's interest closely.

(2) **Dynamic update.** Our scheme supports the dynamic update of the document, that is, add, delete, and modify.

(3) **Security.** The scheme should ensure that the adversary (i.e., the CSP) cannot learn other information, such as the searched keyword and the plaintext information of the documents from the secure index and the encrypted documents except for the restricted information. The information is allowed to be leaked to the CSP includes the search pattern and the access pattern, such as the size of encrypted documents, the length and identification of the documents, and search results, etc..

(4) **Fairness.** A smart contract is adopted to achieve fair payment. Ethereum blockchain ensures that the implementation of the smart contract cannot be modified and denied. The CSP and DU execute honestly according to the rules of contract, then CSP can get the service fee, and DU gets the correct search results. It is fair to the

CSP because a malicious DU cannot enjoy the search service provided by the CSP without paying the service fee. It is fair to the DU because there is no malicious CSP provides the incorrect search results after receiving the service fee. If the malicious CSP provides the wrong results, it cannot pass the verification algorithm, and the DU will receive the deposit as a penalty for the CSP. The DU can obtain effective search results as long as the service fee is paid, and the CSP can obtain the service fee as long as the correct result is returned.

(5) **Reliability.** The scheme uses a verification mechanism to verify the search results, and prevents the CSP from performing the search operation dishonestly.

F. RELIABILITY DEFINITION

Definition 3: Let π denote a dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain and A is an arbitrary Probability Polynomial Time (PPT) adversary. And the CSP is treated as adversary in our paper. We consider the following probability game.

$Forge_A^{\pi}(\lambda)$: Given a search token $T_{w'}$, the adversary forges a collection of search results $C'_{w'}$ and the corresponding verification evidence $pf(C)'$, where $C'_{w'} \neq C_{w'}$, $pf(C)' \neq pf(C)$, and $(C_{w'}, pf(C)) \leftarrow Search(I, C, T_{w'})$. If $Verify(acc(C), pf(C)', C'_{w'}) = C'_{w'}$, the game outputs a bit.

The scheme π satisfies the reliability, if for all PPT adversaries A, the probability of winning the game is negligible, i.e., $|\Pr[Forge_A^{\pi}(\lambda)] = 1| \leq negl(\lambda)$, where $negl(\lambda)$ is a negligible function. When given a valid $C_{w'}$ and $pf(C)$, if the adversary can forge $C'_{w'}$, $pf(C)'$, and pass the *Verify* algorithm, it can win the game. Reliability means that the probability that an adversary forges search results and evidence successfully is negligible.

IV. SPECIFIC CONSTRUCTION

A. SYSTEM INITIALIZATION PHASE

$Setup(1^{\lambda}) \rightarrow (MK, PK)$. Taking security parameter λ as input, the DO randomly generates a pair of secure primes p and q, where $N = pq$. Let G is a cyclic group and g is a generator of G, where the group G is represented as $G = \{u = v^2 \mod N | v \in Z_N^*\}$. The DO generates the public key (N, g) and calls the prime generation function $P(\cdot)$ in [40], [41]. The DO defines a pseudo-random permutation $\pi : \{0,1\}^{\lambda} \times \{0,1\}^l \rightarrow \{0,1\}^l$, where l represents the maximum length of the keyword. It selects a collision-resistant hash function $f : G \rightarrow \{0,1\}^n$ for encrypting index vector, and chooses a collision-resistant hash function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ for calculation of verification. Finally, the setup algorithm outputs system master private keys $MK = (p, q)$ and system public parameters $PK = (N, g, \pi, f, H, P(\cdot))$.

B. KEY GENERATION PHASE

$KeyGen(1^{\lambda}) \rightarrow (sk, k_1)$. With the security parameter λ, the DO generates a symmetric key sk, and randomly selects $k_1 \leftarrow \{0,1\}^{\lambda}$ as the key of π. The key generation algorithm outputs keys sk, k_1, and transmits them to the authorized DU through a secure channel.

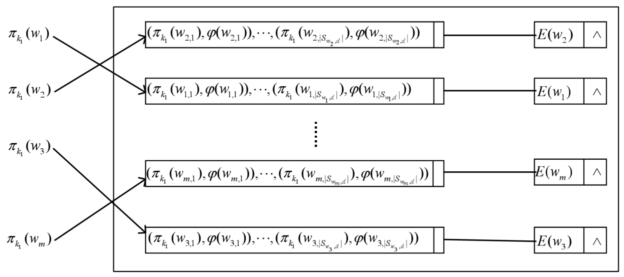
$\pi_{k_1}(w_1)$
$\pi_{k_1}(w_2)$
$\pi_{k_1}(w_3)$
$\pi_{k_1}(w_m)$
$(\pi_{k_1}(w_{2,1}), \varphi(w_{2,1})), \cdots, (\pi_{k_1}(w_{2,|S_{w_2,d}|}), \varphi(w_{2,|S_{w_2,d}|}))$
$(\pi_{k_1}(w_{1,1}), \varphi(w_{1,1})), \cdots, (\pi_{k_1}(w_{1,|S_{w_1,d}|}), \varphi(w_{1,|S_{w_1,d}|}))$
$(\pi_{k_1}(w_{m,1}), \varphi(w_{m,1})), \cdots, (\pi_{k_1}(w_{m,|S_{w_m,d}|}), \varphi(w_{m,|S_{w_m,d}|}))$
$(\pi_{k_1}(w_{3,1}), \varphi(w_{3,1})), \cdots, (\pi_{k_1}(w_{3,|S_{w_3,d}|}), \varphi(w_{3,|S_{w_3,d}|}))$
$E(w_2)$
$E(w_1)$
$E(w_m)$
$E(w_3)$
$\wedge$

FIGURE 2. Index structure.

C. ENCRYPTION PHASE

$$Enc(PK, D, sk, k_1) \rightarrow (C, I, acc(C)).$$

Step 1. Documents encryption. $FileEnc(D, sk) \rightarrow C$. The DO uses sk to encrypt plaintext documents $D_i (i \in [n])$ and obtains $C_i = \{Enc_{sk}(D_i) | i \in [n]\}$. The DO sends the encrypted documents collection $C = \{C_1, C_2, \cdots, C_n\}$ to the CSP.

Step 2. Secure index generation. $IndexGen(PK, D, k_1) \rightarrow I$. Our scheme constructs a secure index by extending the inverted index, as shown in Fig. 2.

The DO scans the plaintext documents collection $D = \{D_1, D_2, \cdots, D_n\}$ and extracts keywords. Let $W_D = \{w_1, w_2, \cdots, w_m\}$ denote the keywords set. For each keyword $w_i \in W$, the DO defines a n-dimension vector $v(w_i)$. If the j-th document contains the keyword w_i, then $v(w_i)[j] = 1$, otherwise $v(w_i)[j] = 0$. For example, suppose there are three files D_1, D_2, and D_3, where D_1 contains keywords w_1 and w_2, D_2 contains keywords w_2 and w_3, as well as D_3 contains keywords w_1 and w_2, we can get $v(w_1) = [101]$, $v(w_2) = [111]$, and $v(w_3) = [010]$.

The DO constructs fuzzy keywords set for each keyword w_i in $W_D = \{w_1, w_2, \cdots, w_m\}$. Let $S_{w_i,d}$ denote fuzzy keywords set of the keyword w_i with edit distance d, and let $w_{i,t} (1 \leq i \leq m, 1 \leq t \leq |S_{w_i,d}|)$ represent a keyword in the fuzzy keyword set $S_{w_i,d}$.

The DO calculates $\pi_{k_1}(w_{i,t})$ and $\varphi(w_{i,t}) = g^{\prod_{w_{i,j} \in S_{w_i,d} - w_{i,t}} \pi_{k_1}(w_{i,j})}$ for each keyword $w_{i,t}$ in $S_{w_i,d}$, where $1 \leq i \leq m$, $1 \leq t \leq |S_{w_i,d}|$, and stores $(\pi_{k_1}(w_{i,t}), \varphi(w_{i,t}))$ on the first node of the inverted index.

For each keyword $w_i \in W_D$, the DO computes $f(\phi(w_i)) \oplus v(w_i) \rightarrow E(w_i)$, where $\phi(w_i) = g^{\prod_{w_{i,j} \in S_{w_i,d}} \pi_{k_1}(w_{i,j})}$. Then DO obtains $E(w_i)$ by encrypting vector $v(w_i)$, and stores the encrypted index vector $E(w_i)$ on the second node of the inverted index.

The DO sends $(\pi_{k_1}(w_{i,t}), \varphi(w_{i,t}))$ and $E(w_i)$ to the CSP as the secure index I.

Step 3. Calculate the accumulated value. $CalAcc(PK, C) \rightarrow acc(C)$. The DO applies a dynamic accumulator on encrypted documents collection $C = \{C_1, C_2, \ldots, C_n\}$. For collection $C = \{C_1, C_2, \ldots, C_n\}$, the DO calculates the accumulated value $acc(C)$: $acc(C) = g^{\prod_{j=1}^{n} P(H(C_j))} \bmod N$. Then DO creates a smart contract via transaction TX_1 and calls the *AddACC* function to store the accumulated value $acc(C)$.

Algorithm 1 *AddACC*

```
    Input: $acc(C)$
    Output: null
1 if msg.sender is not the data owner then
2     throw
3 else acc = $acc(C)$
4 end
```

D. SEARCH TOKEN GENERATION PHASE

TokenGen$(PK, w', k_1) \rightarrow T_{w'}$. When the DU wants to search the documents containing the misspelled keyword w', it calculates the search token $T_{w'} = \pi_{k_1}(w')$ by using the shared key k_1 and sends token to the CSP.

E. SEARCH PHASE

Search$(PK, C, I, T_{w'}) \rightarrow (C_{w'}, pf(C))$. The search operation is performed by the CSP. After receiving the search token $T_{w'}$, the cloud server matches the element of the first node of each linked list with the search token. In our scheme, the encrypted exact keyword $\pi_{k_1}(w_i)$ is used to represent the first element $\pi_{k_1}(w_{i,1})$ of the first node, i.e., $w_{i,1}$ represents an exact keyword w_i. Therefore, the cloud server first determines if the search token $\pi_{k_1}(w')$ equals the first elements $\pi_{k_1}(w_{i,1})(1 \leq i \leq m)$, and then matches the other encrypted keywords $\pi_{k_1}(w_{i,t})(1 < t \leq |S_{w_i,d}|)$ in $S_{w_i,d}$.

When getting the corresponding fuzzy keyword ciphertext through searching, i.e., $\pi_{k_1}(w_{i,t}) = \pi_{k_1}(w')$, the CSP can find the corresponding label $\varphi(w_{i,t})$. Then the CSP obtains $\phi(w_i)$ by calculating $\varphi(w_{i,t})^{\pi_{k_1}(w_{i,t})}$. The CSP decrypts the encrypted index $E(w_i)$ to get the index vector $v(w_i)$ by computing $f(\phi(w_i)) \oplus E(w_i) \rightarrow v(w_i)$. If $v(w_i)[j] = 1$, the CSP adds the encrypted document C_j to the encrypted documents collection $C_{w'}$ which contains the searched keyword.

Let $v(w_i) = (e_1, e_2, \cdots, e_n)$. For encrypted documents collection $C = \{C_1, C_2, \ldots, C_n\}$, CSP calculates verification evidence $pf(C) = g^{\prod_{e_j=0} P(H(C_j))} \bmod N (1 \leq j \leq n)$.

The CSP sends the search results $C_{w'} = \{C_{w',1}, C_{w',2}, \cdots, C_{w',j}\}$ and the verification evidence $pf(C)$ to the smart contract through a transaction TX_2, and submits the deposit \$ Guaranty which is transferred to the smart contract for temporarily storing. These processes can be completed by calling the *AddProof* function in the smart contract. When the verification algorithm fails, i.e., the CSP returns error results, the CSP will lose the deposit.

F. VERIFICATION PHASE

Verify$(PK, C_{w'}, acc(C), pf(C)) \rightarrow C_{w'}/\perp$. The DU temporarily stores the service fee Fee \$ to the smart contract via transaction TX_3 and calls the *Verify* function in the contract. The smart contract verifies whether the search results returned by the cloud server is correct, and solves the service-payment fairness problem. The smart contract allows a trusted transaction without a third party. When the verification passes, the contract returns $C_{w'} = \{C_{w',1}, C_{w',2}, \cdots, C_{w',j}\}$ to the DU, the CSP obtains the service fee and redeems the deposit. When the verification fails, the DU obtains the deposit and redeems the service fee.

Algorithm 2 *ADDProof*

```
    Input: $C_{w'}$, $pf(C)$, $ Guaranty
    Output: null
1 if msg.sender is not server or msg.value <$ Guaranty then
2     throw
3 else cipher [] = $C_{w'} = \{C_{w',1}, C_{w',2}, \ldots, C_{w',j}\}$, pf = $pf(C)$
4     send $C$ Guaranty to contract
5 end
```

Algorithm 3 *Verify*

```
    Input: $C_{w'}$, $Pf(C)$, $ Fee
    Output: null or cipher[]
1 if msg.sender is not the data user or msg.value < $ Fee then
2     throw
3 else send $ Fee to contract
4 for $C_j$ in $C_{w'}$ do
5     $x_j = P(H(C_j))$
6 if $acc(C) = pf(C)^{\prod_{j \in C_{w'}} x_j}$ then
7     return cipher[]
8     send contract.balance to CSP
9 else send contract.balance to user
10 end
```

G. DECRYPTION PHASE

Dec$(sk, C_{w'}) \rightarrow D_{w'}$. The DU decrypts the encrypted documents $C_{w'} = \{C_{w',1}, C_{w',2}, \cdots, C_{w',j}\}$ returned by the contract through the symmetric key sk, and obtains the closest plaintext documents $D_{w'} = \{AES.Dec(sk, C_{w',i}) | i \in [1, j]\}$ containing the searched keyword w'.

H. UPDATE PHASE

Update$(PK, acc(C), D_{n+1}, D_j) \rightarrow (C', I', acc(C)')$. In practical application, DO can dynamically and arbitrarily add, delete, or modify the document.

Case 1: Add document. The DO adds a document D_{n+1}, then encrypts the document D_{n+1} using the *FileEnc* algorithm, and calls *IndexGen* algorithm to update the secure index I'. Let $n = n+1$. The DO sends the encrypted document C' to the CSP along with the encrypted index I'. The DO calculates the accumulated value after updating: $acc(C)' = acc(C)^{P(H(C_{n+1}))} \bmod N$.

Case 2: Delete document. The DO tends to delete the document D_j, it calculates the ciphertext C_j, and updates the secure index I' by calling the *IndexGen* algorithm. Let $n = n - 1$. Then DO sends I' and C_j to the CSP. After receiving the DO's deletion request, the CSP deletes C_j in the encrypted documents collection C. Finally, the DO calculates $x_j = P(H(C_j))$ and obtains the updated accumulated value $acc(C)' = acc(C)^{(x_j)^{-1}} \bmod N$.

Case 3: Modify document. The DO wants to convert a document D_j to D'_j. It gets C'_j by encrypting D'_j, and calls *IndexGen* algorithm to generate an updated index I'. After receiving the DO's modification request, CSP replaces C_j with C'_j in encrypted documents collection C. DO calculates $x_j = P(H(C_j))$, $x'_j = P(H(C'_j))$, and obtains the updated accumulated value $acc(C)' = acc(C)^{(x_j)^{-1} \cdot x'_j} \bmod N$.

As the smart contract cannot be modified once deployed, we need to redeploy the update contract if we want to update the accumulated value in the contract. After updating the document, DO needs to deploy the update contract through transaction TX_4 and stores the updated accumulated value $acc(C)'$. The update smart contract still includes algorithm 2 and algorithm 3. Here we only list the algorithm 4, which is different from those in the verification smart contract.

Algorithm 4 *UPdate*

```
    Input: $acc(C)'$
    Output: null
1 if msg.sender is not the data owner then
2   throw
3 else acc = $acc(C)'$
4 end
```

V. SCHEME ANALYSIS

A. CORRECTNESS ANALYSIS

Dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain is correct in calculation.

In the search phase, after getting the corresponding fuzzy keyword ciphertext through searching the index I, i.e., $\pi_{k_1}(w_{i,t}) = \pi_{k_1}(w')$, the CSP can find the corresponding label $\varphi(w_{i,t})$. The CSP calculates $\varphi(w_{i,t})^{\pi_{k_1}(w_{i,t})}$ to get $\phi(w_i)$ with the formula $\varphi(w_{i,t})^{\pi_{k_1}(w_{i,t})} = (g^{\prod_{w_{i,j} \in S_{w_{i,d}} - w_{i,t}} \pi_{k_1}(w_{i,j})})^{\pi_{k_1}(w_{i,t})} = g^{\prod_{w_{i,j} \in S_{w_{i,d}}} \pi_{k_1}(w_{i,j})} = \phi(w_i)$.

In the verification phase, the smart contract enters the search results $C_{w'} = \{C_{w',1}, C_{w',2}, \cdots, C_{w',j}\}$, verification evidence $pf(C)$, and the accumulated value $acc(C)$, and calculates

$$\begin{aligned} & pf(C)^{\prod_{j \in C_{w'}} P(H(C_j))} \bmod N \\ & = (g^{\prod_{e_j=0} P(H(C_j))})^{\prod_{j \in C_{w'}} P(H(C_j))} \bmod N \\ & = g^{\prod_{j=1}^{n} P(H(C_j))} \bmod N = acc(C), \end{aligned}$$

which indicates that the CSP has no malicious behavior and the search results $C_{w'} = \{C_{w',1}, C_{w',2}, \cdots, C_{w',j}\}$ are correct.

B. SECURITY ANALYSIS

Theorem 1: If the symmetric encryption algorithm is semantic security, and π is a pseudo-random permutation, f, H are two collision-resistance hash functions, then our proposed fuzzy keyword searchable encryption scheme based on blockchain is non-adaptive semantic secure.

Proof: If for any PPT adversary A, there is a simulator S, which satisfies the following condition $|\Pr[V(H) \leftarrow Real_A^{\pi}(\lambda)] - \Pr[V^*(H^*) \leftarrow Sim_{A,S}^{\pi}(\lambda)]| \leq negl(\lambda)$, where $negl(\lambda)$ is a negligible function, in other words, the outputs of $Real_A^{\pi}(\lambda)$ and $Sim_{A,S}^{\pi}(\lambda)$ are indistinguishable, then our scheme is non-adaptive semantic secure. The adversary A will win the game by analyzing the ciphertext, index, and search token generated by the simulator S. S can simulate the view $V^*(H^*)$ according to the $Tr(H^*)$ such that the CSP unable to distinguish $V(H)$ and $V^*(H^*)$, where $Tr(H^*) = Tr(H)$.

Simulator S is able to reach this objective by the following procedures. When given $Tr(H) = Tr(H^*) = (n = |D|, \{|D_i|, id(D_i)\}_{i \in [n]}, m = |W_D|, |S_{w_i,d}|_{i \in [m]}, |D_{n+1}|, |D_j|, id(D_{n+1}), id(D_j), AP(W), SP(Q))$, the process of the simulator S generates the simulated ciphertext documents, the simulated index, the simulated search token, the simulated updated ciphertext document, and the simulated updated index are as follows:

(1) Simulate encrypted documents C^*: According to the trace of history $Tr(H^*) = Tr(H)$, the simulator S randomly generates n simulated encrypted documents with the length of $|D_i|_{i \in [n]}$ bits, i.e., $C^* = \{C_i^* \in \{0, 1\}^{|D_i|}\}_{D_i \in D, i \in [n]} = \{C_1^*, C_2^*, \cdots, C_n^*\}$.

(2) Simulate secure index I^*: The simulator sets I^* as a look-up table with m entries and each entry has three nodes. For the first node, S randomly selects m elements a_i^* in $\{0, 1\}^l$, where $i \in [m]$. To generate second node, the simulator S randomly generates $|S_{w_{i,d}}|$ elements $b_{i,t}^*$ in $\{0, 1\}^l$, $d_{i,t}^*$ in G for each keyword a_i^*, where $i \in [m], 1 \leq t \leq |S_{w_{i,d}}|$. The simulated second node is used to replace the $\pi_{k_1}(w_{i,t})$ and $\varphi(w_{i,t})$ for each fuzzy keyword $w_{i,t}$ in $S_{w_{i,d}}$ of the same length with random strings $b_{i,t}^*$ and $d_{i,t}^*$, where $1 \leq i \leq m, 1 \leq t \leq |S_{w_{i,d}}|$. The simulator stores $(b_{i,t}^*, d_{i,t}^*)_{1 \leq i \leq m, 1 \leq t \leq |S_{w_{i,d}}|}$ on the second node of the index I^*. According to the trace of history $Tr(H) = Tr(H^*) = (n = |D|, \{|D_i|, id(D_i)\}_{i \in [n]}, m = |W_D|, |S_{w_i,d}|_{i \in [m]}, |D_{n+1}|, |D_j|, id(D_{n+1}), id(D_j), AP(W), SP(Q))$, and $AP(W)$ which is expressed as $(id(D_1), \cdots, id(D_{|D(w_i)|}), |D_1|, \cdots, |D_{|D(w_i)|}|)_{i \in [m]}$, the simulator S generates an encrypted vector $E(a_i^*)^*$ for each keyword a_i^* and stores them on the third node, where $i \in [m]$. The simulated index I^* is $(a_i^*, (b_{i,t}^*, d_{i,t}^*)_{1 \leq i \leq m, 1 \leq t \leq |S_{w_{i,d}}|}, E(a_i^*)^*)$. The process of constructing an index in $Real_A^{\pi}(\lambda)$ by using pseudo-random permutation π, collision-resistance hash functions f, H. The simulated index I^* is used to replace the $\pi_{k_1}(w_i)$, $\pi_{k_1}(w_{i,t})$, $\varphi(w_{i,t})$, and $E(w_i)$ of the same length with random strings $(a_i^*, (b_{i,t}^*, d_{i,t}^*), E(a_i^*)^*)_{1 \leq i \leq m, 1 \leq t \leq |S_{w_{i,d}}|}$.

(3) Simulate search token T_Q^*: The simulator S randomly selects q elements in $\{0, 1\}^l$ as the simulated search token $T_{w_i}^*$, where $i \in [1, q]$.

(4) Simulate updated C'^*, I'^*: In the update phase, three operations are involved, i.e., addition, deletion and modification. After updating is completed, we need to update the related elements in $Tr(H^*) = Tr(H)$, such as n, m, etc.. In the process of adding file D_{n+1}, deleting or modifying file D_j, the simulator S randomly generates a simulated encrypted document C'^* with the length of $|D_{n+1}|or|D_j|$ bits, i.e., $C'^* \in \{0, 1\}^{|D_{n+1}|or|D_j|}$ according to the updated trace of history $Tr(H^*) = Tr(H)$. At the same time, the simulator S uses the method (2) of simulating secure index to generate the updated security index I'^* according to the updated trace of history $Tr(H^*) = Tr(H)$.

Because of the symmetric encryption algorithm is semantic security, it can be guaranteed that C of $Real_A^\pi(\lambda)$ and C^* in $Sim_{A,S}^\pi(\lambda)$ are computationally indistinguishable without obtaining the symmetric key sk, that is, $|\Pr[C \leftarrow Real_A^\pi(\lambda)] - \Pr[C^* \leftarrow Sim_{A,S}^\pi(\lambda)]| \le negl_1(\lambda)$. Similarly, semantic security of symmetric encryption can ensure that C' of $Real_A^\pi(\lambda)$ and C'^* in $Sim_{A,S}^\pi(\lambda)$ are computationally indistinguishable without obtaining the symmetric key sk, that is, $|\Pr[C' \leftarrow Real_A^\pi(\lambda)] - \Pr[C'^* \leftarrow Sim_{A,S}^\pi(\lambda)]| \le negl_2(\lambda)$. The probability of the simulator generates a valid search token is negligible without allocating k_1. For $\forall w_i \in Q$, the search token is defined as $T_{w_i} = \pi_{k_1}(w_i)$, and π is a pseudo-random permutation. Since the simulator S does not know k_1, the probability of selecting k_1^* randomly and constructing a valid search token is $\Pr[T_{w_i}^* = T_{w_i}; T_{w_i}^* = \pi_{k_1^*}(w_i); T_{w_i} = \pi_{k_1}(w_i); k_1^* \leftarrow \{0, 1\}^\lambda] \approx \Pr[k_1 = k_1^* | k_1^* \leftarrow \{0, 1\}^k] \approx \frac{1}{2^k}$. Therefore, we can know from the definition of the negligible function in [3], for the pseudo-random permutation π, the indistinguishable between the search token in $Real_A^\pi(\lambda)$ and the search token in $Sim_{A,S}^\pi(\lambda)$ can be guaranteed in the case where the search key k_1 is not known, the indistinguishability of the search token is based on the indistinguishability between the output of pseudo-random permutation and random string, that is, $|\Pr[T_Q \leftarrow Real_A^\pi(\lambda)] - \Pr[T_Q^* \leftarrow Sim_{A,S}^\pi(\lambda)]| \le negl_3(\lambda)$. For the same reason, the indistinguishability of the index is also based on the indistinguishability between the output of pseudo-random permutation and random string of the same size without knowing the key k_1. Therefore, the adversary cannot distinguish I and I^*, I' and I'^*, that is, $|\Pr[I \leftarrow Real_A^\pi(\lambda)] - \Pr[I^* \leftarrow Sim_{A,S}^\pi(\lambda)]| \le negl_4(\lambda)$, $|\Pr[I' \leftarrow Real_A^\pi(\lambda)] - \Pr[I'^* \leftarrow Sim_{A,S}^\pi(\lambda)]| \le negl_5(\lambda)$.

In conclusion, for any PPT adversary A, the outputs of $Real_A^\pi(\lambda)$ and $Sim_{A,S}^\pi(\lambda)$ are indistinguishable, i.e., $|\Pr[V(H) \leftarrow Real_A^\pi(\lambda)] - \Pr[V^*(H^*) \leftarrow Sim_{A,S}^\pi(\lambda)]| \le negl(\lambda)$. Therefore, our scheme satisfies non-adaptive semantic security.

C. RELIABILITY ANALYSIS

Theorem 2: Dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain meets reliability.

Proof: Suppose there is a PPT adversary A who can break the reliability of our scheme with a non-negligible probability. According to this assumption, the adversary is able to forge the search result $C'_{w'}$ and the corresponding verification evidence $pf(C)'$ in order to satisfy $Verify(acc(C), pf(C)', C'_{w'}) = C'_{w'}$. If such an adversary exists, it will violate the strong RSA security of the dynamic accumulator in [40], [41]. The hypothesis $(C_{w'}, pf(C))$ is the correct search results, it is necessary to prove that the adversary A cannot forge valid search results to make the equation $(C'_{w'}, pf(C)') = (C_{w'}, pf(C))$ true. We consider the following four cases:

(1) $C'_{w'} \neq C_{w'}, pf(C)' \neq pf(C)$. The adversary A calculates $acc(C)' = pf(C)'^{\prod_{j \in C'_{w'}} P(H(C_j))} \bmod N$, if it does not equal $acc(C)$, verification fails.

(2) $C'_{w'} \neq C_{w'}, pf(C)' = pf(C)$. The adversary A calculates $acc(C)' = pf(C)^{\prod_{j \in C'_{w'}} P(H(C_j))} \bmod N$, if it does not equal $acc(C)$, verification fails.

(3) $C'_{w'} = C_{w'}, pf(C)' \neq pf(C)$. The adversary A calculates $acc(C)' = pf(C)'^{\prod_{j \in C_{w'}} P(H(C_j))} \bmod N$, if it does not equal $acc(C)$, verification fails.

(4) $C'_{w'} = C_{w'}, pf(C)' = pf(C)$. The adversary A calculates $acc(C)' = pf(C)^{\prod_{j \in C_{w'}} P(H(C_j))} \bmod N$, even if it equals $acc(C)$, the strong RSA hypothesis guarantees that no one can find another set C' which is different from C and the accumulated value satisfies $acc(C') = acc(C)$ in the probability polynomial time without knowing p and q, so the verification fails.

In the above four cases, if the search results $C'_{w'}$ and evidence $pf(C)'$ that forged by adversary A enable the verification algorithm pass, then the strong RSA security of the dynamic accumulator is broken. Because of the unidirectionality and unforgeability features of the dynamic accumulator in [40], the probability that the adversary forges valid search results and evidence is negligible, so it is contrary to the assumption. Therefore, there is no PPT adversary A can break the reliability of the scheme with a non-negligible probability. The dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain satisfies reliability.

D. FAIRNESS ANALYSIS

The reliability of the smart contract ensures the fair exchange of our proposed scheme between the cloud server and the user. Smart contract performs a predefined verification algorithm.

Due to the features of the ethereum blockchain, such as irreversible, undeniable, and traceable, CSP and DU are constrained to execute honestly according to the rules of the contract. The DU can get the correct encrypted documents

TABLE 3. Function comparison.

Scheme	Blockchain-based	Keyword type	Fair payment	Verification side	Update dynamically
Scheme [28]	√	Exact keyword	√	User node	×
Scheme [32]	√	Exact keyword	√	Smart contract	√
Scheme [37]	×	Fuzzy keyword	×	User node	×
Our scheme	√	Fuzzy keyword	√	Smart contract	√

$C_{w'} = \{C_{w',1}, C_{w',2}, \cdots, C_{w',j}\}$ without needing additional local verification. We use the verification contract to verify the correctness of search results returned by the CSP, so our scheme reduces the user's computational burden.

Because of the irreversible feature of blockchain, the smart contract cannot be modified once it deployed. Only if the DU gets the correct results, the CSP can get the service fee and redeem deposit. Only if the DU pays the service fee, then it can obtain the correct results. If the CSP dishonestly returns the error or incomplete results, the verification contract will find the malicious behavior, return the service fee to the DU, and the DU receives the deposit as a penalty for the CSP. In addition, the DU cannot deny the search service provided by the CSP and refuse to pay the service fee if the returned search results are verified to be correct. Because the DU needs to generate the transaction by using a unique externally owned account and the service fee has been locked in the contract. Once the transaction is confirmed and recorded on the blockchain, it cannot be modified or denied. Therefore, CSP returns the correct search results honestly in order to obtain the service fee.

We introduce the blockchain fairness mechanism into the searchable encryption scheme to achieve service-payment fairness.

E. COMPARISON ANALYSIS

1) FUNCTION COMPARISON

Comparing the function features of our proposed scheme with the literature [28], [32], [37] from the aspects of blockchain-based, supporting the type of keyword search, fair payment, verification side, and whether supports update dynamically, as shown in Table 3. Both the scheme [28] and the scheme [32] are searchable encryption schemes that support exact keyword search, and use blockchain technology to achieve service-payment fairness. But the scheme of [28] in which the search results are verified by the user, increasing the computational cost of the user. In addition, to reduce the user's local computational overhead, both the literature [32] and the proposed scheme use smart contract to verify the correctness of the search results. However, the literature [32] only supports exact keyword retrieval, which requires the keyword inputted by the user must exactly match the predefined keywords in the index. Furthermore, both our scheme and the scheme in [37] support fuzzy keyword retrieval, but the scheme in [37] does not have the functions of fair payment and dynamic update of the document, meanwhile, the user verifies the correctness and integrity of the search results locally.

From the function point of view, combining with blockchain and smart contract technology, our scheme supports verifiable fuzzy keyword search, ensures the security and correctness of search results, and achieves service-payment fairness. The user can get the correct search results only by paying a certain service fee without local verification, so the scheme can reduce the user's computational burden, and is more in line with the cloud storage application.

2) PERFORMANCE COMPARISON

Comparing the computational cost of our scheme with related schemes in the generation of the index and search token, the efficiency of search and verification, comparing the number of transactions for the blockchain-based searchable encryption scheme, and comparing the times of the search token is submitted, as shown in Table 4.

In the index generation phase, the scheme in [28], [32] supports the exact keyword search. The scheme [28] needs to calculate $mjP + mn(H + SIG)$, so the index generation cost is related to the number of the keywords, the number of documents containing keywords and the total number of the documents. And the public key signature algorithm which generally has a relatively large amount of calculation is used. So it can be seen that the computational cost of the scheme [28] in the index generation phase is higher. The index generation cost of the scheme [32] is only related to the number of keywords and the number of documents containing keywords. Because their scheme only requires 1 pseudo-random function operation and 1 hash operation, its computation amount is smaller than [28]. Both our scheme and scheme [37] support the fuzzy keyword search, but the scheme [37] needs to calculate n times the symmetric encryption algorithm. Since the total number of documents n is generally large, its calculation cost is much higher. From the point of view of the result analysis, the computational cost of our scheme in the index generation phase is relatively minimal.

In the process of search token generation and submission, the time to generate a search token increases linearly with the number of documents increases in the scheme [28]. The computational cost of the search token generation of both schemes [32], [37] and our scheme are at a constant level. In [32], the searched keyword performs 1 pseudo-random function, and the number of search token submitted is twice, which one of the search tokens is submitted to the CSP and the other search token is submitted to the smart contract. Scheme [37] needs to calculate 1 pseudo-random permutation

TABLE 4. Performance comparison.

Scheme	Index generation phase	Search token generation phase	Search phase	Verification phase	Number of transactions	Number of token submitted	Update complexity
Scheme [28]	$mjP+mn(H+SIG)$	nP	$O(\|D(w)\|)$	$n(H+SIG)$	13	1	-
Scheme [32]	$2mF+jH$	$1F$	O(1)	$\|D(w)\|^2$	6+n	2	$O(m')$
Scheme [37]	$ms(P+H)+m(F+XOR)+nSKE$	$1P+1F$	$O(ms)$	$1XOR+1F$	-	2	-
Our scheme	$ms(P+E)+m(E+XOR)$	$1P$	$O(ms)$	$1E+2H\|D(w)\|$	4	1	$O(m')$, $O(1)$

Notes: m indicates the number of keywords, n indicates the number of documents, j indicates the number of documents which contains keyword, m' indicates the number of keywords in the updated document, $|D(w)|$ indicates the number of documents that contains searched keyword, s indicates the size of the fuzzy keywords set, P indicates the pseudo-random permutation, F indicates a pseudo-random function, XOR stands for an exclusive-OR operation, E means exponential operation, H means hash operation, SIG means signature algorithm which includes processes of signature and verification, SKE means symmetric encryption algorithm, - indicates that this operation does not need.

and 1 pseudo-random function for the searched keyword. When the search token is sent to the CSP, it needs to be submitted again if the first node in the secure index does not match successfully, so the search token is submitted twice. Our scheme calculates 1 pseudo-random permutation for the searched misspelled keyword and needs to submit the search token to the CSP only once. As you can see, in the search token generation and submission phase, our scheme also has an advantage.

In the search phase, the scheme [28] uses an inverted index structure, and the complexity of the search is linearly related to $|D(w)|$. In scheme [32], because of the index structure bases on a key-value pair, the search complexity is O(1). Both the scheme [37] and our scheme support the fuzzy keyword search. When searching for the related documents, the CSP matches the encrypted value of the first element of the first nodes in the index, and then matches the rest of fuzzy keywords in the fuzzy keywords set, so the search efficiency is related to ms.

In the verification stage of the search results, the scheme [28] requires the user to perform n times hash operations and signature verification operations. With the increasing of the number of the returned documents, the workload of verification for the user will greatly increase. The verification process in [32] is performed by the smart contract, and the hash value of the encrypted documents which contains the queried keyword is matched $|D(w)|^2$ times. The verification process of the scheme [37] is completed by the user, and $1XOR$ and 1 hash operation is calculated on the search results to determine the results correctly or not. Our scheme executes 1 exponential operation and $2|D(w)|$ times hash operations. But it adopts the smart contract for verification to reduce the computational overhead of the user.

Both our scheme and scheme [28], [32] are all based on blockchain technology. The scheme [28] needs to conduct 13 transactions. The scheme [32] needs to conduct $6+n$ transactions. But the number of transactions in our scheme is only 4. Obviously, our scheme reduces the delay in the processes of uploading, packing, and confirming of the transactions on the blockchain, making the system has a high response speed.

Besides, both our scheme and scheme [32] support dynamic update of file. Let m' represent the number of keywords in the updated document. When adding a file, the scheme [32] needs to perform $m'(F+H)$ operations. When deleting a file, the scheme [32] needs to calculate the search token for the m' keywords through F operation, and then uses the token to find the corresponding index entries and delete them. Therefore, the update complexity of scheme [32] is $O(m')$, the efficiency of updating is related to m'. When adding a file, our scheme needs to update the corresponding index vector for keywords that already exist in the index. For the newly generated keywords, our scheme needs to calculate $s(P+E)+E+XOR$ to generate new index entries. When DO updates the accumulated value, it needs to calculate $1H+1E$. So the update efficiency of the accumulated value is $O(1)$. When deleting a file, our scheme needs to update the index vector related to the keywords contained in the file, and changes the vector related to the deleted file in the index vector from 1 to 0. When DO updates the accumulated value, it also needs to calculate $1H+1E$. When modifying a file, our scheme needs to modify the index vector related to the keywords contained in the file, and DO needs to calculate $s(P+E)+E+XOR$ to generate new index entries for the newly added keywords. The accumulated value needs to be updated by calculating $2H+1E$. In a word, the update efficiency of index of our scheme is $O(m')$ which is related to m', and the update efficiency of the accumulated value is $O(1)$.

In conclusion, in terms of function, our paper implements fuzzy keyword search, uses smart contract to verify the correctness of the search results, and supports data update dynamically. At the same time, the performance of our scheme is more efficient than other schemes in the processes of index generation, search token generation, and the times of submission of the search token. The efficiency in the search phase is the same as the scheme [37] which both have the same function of fuzzy keyword search. Therefore, on the whole, our scheme has been optimized in terms of performance under the premise of prominent functions, so it is more suitable for the ciphertext retrieval application in the cloud environment.

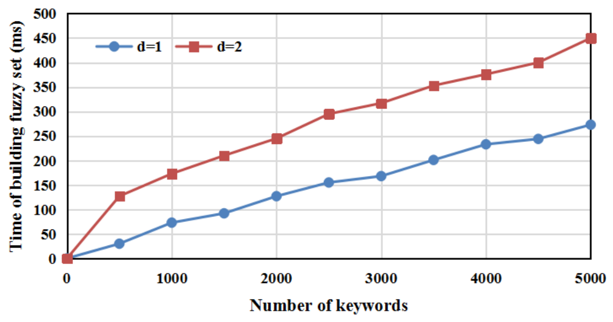

FIGURE 3. Fuzzy keywords set generation time.

F. EXPERIMENT ANALYSIS

In order to more accurately evaluate the actual performance of the proposed scheme, our paper uses the real data set to simulate the fuzzy keywords set generation time, index generation time, search token generation time, search time, and verification time. Our experiment uses papers in the IEEE database as a test data set, and uses Java library of PDFBox to extract keywords from PDF documents. We extract about 5000 keywords from 3000 documents. The experiment adopts Java language, the hash algorithm adopts SHA-256, the pseudo-random permutation adopts HMAC-SHA256, the symmetric encryption algorithm adopts AES-256, the RSA accumulator is used to generate verification evidence, and the smart contract adopts solidity language which runs on the Ethereum virtual machine, the blockchain is achieved based on the Ethereum official test network Rinkeby, and the account uses the ethereum wallet Geth. The hardware environment of this experiment is Intel(R) Core(TM) i5-4200 CPU (2.3GHz), and the RAM is 4GB. We set the parameters in the experiment: the number of documents $n \in [0, 3000]$, the number of keywords $m \in [0, 5000]$, the number of fuzzy keywords $s \in [0, 10000]$, and the number of documents that contain the searched keyword $|D(w)| \in [0, 500]$.

1) INDEX GENERATION PHASE

In the index generation phase, our scheme first needs to build the fuzzy keywords set for each keyword. In the case where the edit distances are respectively $d = 1$ and $d = 2$, the time overhead for generating the fuzzy keywords set is shown in Fig. 3. Both scheme [37] and our scheme use edit distance to generate fuzzy keywords set, so the time cost of generating fuzzy keywords set is the same. Since the length of each keyword is different, the number of fuzzy keywords and the time to generate the fuzzy keywords set are also different. In addition, under the condition that the edit distance is constant, the time overhead of generating the fuzzy keywords set increases with the increase of the number of keywords. Meanwhile, the time cost of the edit distance with 2 is larger than the edit distance with 1.

After generating the fuzzy keywords set, we need to build a secure index. On each fuzzy keyword, we encrypt the fuzzy keyword $\pi_{k_1}(w_{i,t})$, generate label $\varphi(w_{i,t})$, and confuse index vector $E(w_i)$. Therefore, the index generation time depends on the number of keywords. As shown in Fig. 4, when the number of documents is fixed at 3000 and the edit distance is 1, the index generation time increases as the number of keywords increases. In the index generation phase of scheme [37], the time of *nSKE* needs to be calculated. However, in the experimental analysis part, we do not consider the time of file encryption and only compare the generation time of index table between our paper and scheme [37]. Compared with scheme [37], the time of computation of our index table is more expensive, because the time cost of exponential operation is longer than that of hash operation and pseudo-random function.

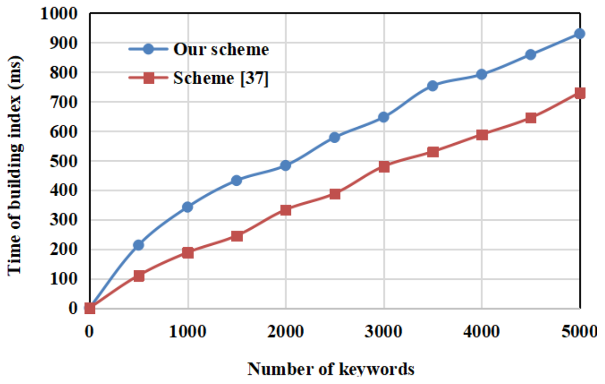

FIGURE 4. Index generation time.

2) SEARCH TOKEN GENERATION PHASE

When DU searches the documents that contain the misspelled keyword w', our scheme needs to compute $T_{w'} = \pi_{k_1}(w')$ by calling HMAC-SHA256 once in the search token generation stage. In our experiment, we want to search for keyword "fuzzy", but we enter the misspelled keyword "fazzy" to test the search token generation time. As shown in Fig. 5, the search token generation time does not change with the increase of the number of the documents, and the average test time is 77ms. In reference [37], one pseudo-random permutation and one pseudo-random function need to be calculated in the stage of generating the search token. In the experiment, HMAC-SHA256 is called twice to complete these calculations and the average generation time is about 150ms.

3) SEARCH PHASE

After receiving the search token $T_{w'}$, the cloud server matches the ciphertext of each fuzzy keyword in the index, then it extracts the corresponding label $\varphi(w_{i,t})$ and decrypts the encrypted index vector $E(w_i)$. If $v(w_i)[j] = 1$, the CSP adds the encrypted document C_j to the documents collection $C_{w'}$ that contains the searched keyword. Therefore, the search complexity is $O(ms)$. The number of keywords and fuzzy keywords will affect search efficiency. Because the number of fuzzy keywords depends on the number of keywords, the search efficiency is related to the number of keywords. In reference [37], the search process still includes three steps:

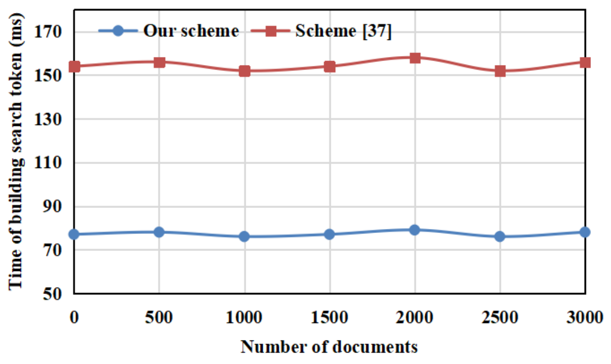

FIGURE 5. Search token generation time.

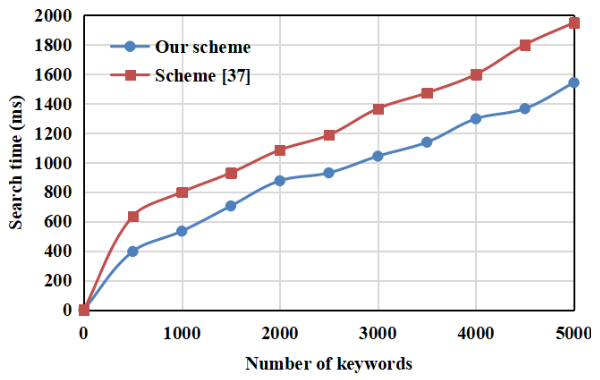

FIGURE 6. Search time.

comparing the search token and fuzzy keywords in the index, extracting the corresponding index vector and verification label, and then decrypting the index vector to obtain the documents containing the queried keyword. However, scheme [37] requires the user and the cloud server to interact with the search token, so the time is increased. As shown in Fig. 6, with the number of keywords increases, the search time of both schemes also increases.

4) VERIFICATION PHASE

In the verification process, we need to calculate 1 exponential operation and 2 hash operations on each document that contains the searched keyword. Therefore, as shown in Fig. 7, with the number of returned documents increases, the verification time also increases. In the verification stage, scheme [37] needs an exclusive-or operation and a hash operation. The implementation of these operations is very fast, so the verification time of scheme [37] will be lower than that of our scheme, and the verification process of reference [37] is completed by the user. Although the verification time of our scheme is relatively high, the verification process is completed by the smart contract, which will not increase the calculation burden of the user, and can achieve fair payment at the same time.

Based on the above analysis, the experiment test and theoretical performance analysis of the proposed scheme are consistent. Compared with other schemes in terms of function and performance, the proposed scheme realizes the functions of file update dynamically, search results verification, fuzzy keyword search, and fair payment at the cost of less performance loss. Although our scheme is not optimal in performance, we extend the function of blockchain-based searchable encryption scheme. Searchable encryption based on blockchain is normally used to verify the logs of search activities, so that unsolicited activities can be tracked and the problem of mutual distrust between the cloud server and the user can be solved. However, most of the existing schemes focus on the exact keyword search. When the keyword searched by the user is misspelled accidentally, the exact keyword search cannot return the correct results. Therefore, a fuzzy keyword searchable encryption based on blockchain, that supports the dynamic storage mechanism and verifiability of the results, is proposed, which greatly improves the usability of the searchable encryption system.

5) COST OF SMART CONTRACT

In ethereum, the deployment and invocation of smart contract require transaction cost. Gas is a special unit used in ethereum. It measures the amount of work required to perform one or a set of operations. Each transaction or contract operation on the ethereum platform needs to consume a certain amount of gas. The more computing resources are consumed, the more gas is consumed. The gas consumption of smart contracts is shown in Table. 5. When experimenting, the transaction price of ethereum is $1\ ether = 199USD$. Let gas price is $1 gasprice = 2 \times 10^{-9} ether$. As shown in Table. 5, the costs of deploying verification contract and updating contract are 0.5192 USD and 0.5384 USD, respectively. The *addACC* function is called by DO to add the accumulated value to the contract. The *addProof* function is called by CSP to add ciphertext documents and verification evidence. The *update* function is called by DO to update accumulated value. The costs of these three functions are 0.0122 USD, 0.0232 USD, and 0.0118 USD, respectively. The *verify* function is called by DU to verify the correctness of search results and achieve fair payment. Because each returned file needs to be calculated, the computational cost of verification increases as the number of returned files increase. Due to the limitation of account balance, we only test the execution cost from 0.0655 USD to 0.4483 USD when the returned file is from 1 to 30.

According to the analysis, the above experimental results show that the gas consumption of the smart contract is acceptable to all participants. Using smart contract to verify the correctness of search results returned by the CSP is feasible.

G. ACCURACY ANALYSIS

Exact keyword search requires that the keywords queried by user are the same as those stored in the index, and then the documents containing the queried keyword are returned accurately. However, unlike exact keyword search, fuzzy keyword search allows user to search for misspelled keyword, and then returns all documents that are as relevant as possible to the queried keyword. Therefore, there is an error between the

TABLE 5. Cost of smart contract.

Function	Number of returned files	Gas used	Ether cost	USD
deploy *verfify* contract	-	1304753	0.002609506	0.5192
deploy *update* contract	-	1352873	0.002705746	0.5384
addACC	-	30865	0.000061730	0.0122
addProof	-	58294	0.000116588	0.0232
verify	1	164721	0.000329442	0.0655
	5	318572	0.000637144	0.1267
	15	495038	0.000990076	0.1970
	20	671346	0.001342692	0.2671
	25	934915	0.001869830	0.3720
	30	1126430	0.002252860	0.4483
update	-	29865	0.000059730	0.0118

returned documents and the user's input interest in the fuzzy keyword search, which is caused by the fuzzy keywords set. In our paper, the fuzzy keywords set is generated by using edit distance. For example, when the keyword queried by the user is "comput", it is a misspelled keyword. In the fuzzy keyword search, the system will return documents containing the keywords "compute", "computer" as relevant as possible. When the keyword searched by the user is "secure", the system will not only return documents containing the keyword "secure", but also return documents containing the keywords "security", "secret", etc.. Because "security" is included in the fuzzy keywords set of "security" and "secret". Therefore, we need to consider the accuracy of the proposed fuzzy keyword search scheme.

In order to evaluate the accuracy of our scheme, we refer to the literatures [38], [45], [46], [49]–[52] and use two basic evaluation criteria that most commonly used in the field of fuzzy keyword search: Precision and Recall. The precision is the ratio of the number of relevant documents in the search results to the total number of documents in the results. The recall is the ratio of the number of relevant documents in the search results to the number of all relevant documents in the documents collection. The precision rate indicates how many of the retrieved documents exactly match the user's input interest. The recall rate indicates how many of all the accurate items in the system have been retrieved. We define *tp* as the number of relevant documents in the search results, and define *fp* as the number of irrelevant documents in the search results. We let *fn* represent the number of relevant documents in unreturned documents. Therefore, the Precision and Recall rate can be calculated by the following formula respectively:

$$Precision = \frac{tp}{tp + fp}$$
$$Recall = \frac{tp}{tp + fn}$$

Fig. 8 shows the precision of our scheme. When the number of returned documents is 100, 200, 300, 400, and 500, the number of relevant documents in the search results is about 94, 171, 243, 292, and 318, respectively. The precision rate is 94%, 85.5%, 81%, 73%, and 63.6%, respectively. The precision represents the ratio of the number of related documents in search results to the number of returned documents. We can see from Fig. 8, when the number of documents in the search results increases from 100 to 500, the precision rate decreases from 94% to 63.6%. When the number of returned documents is small, most of the returned documents are the most relevant documents to the user's search request, so the precision rate is high. However, with the increase in the number of returned documents, more and more unrelated documents are returned, resulting in a downward trend in the precision rate of the search results.

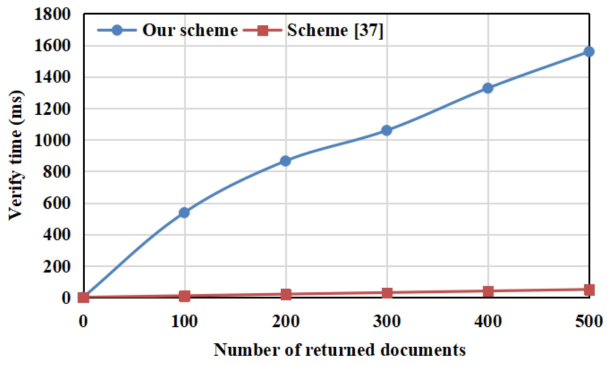

FIGURE 7. Verification time.

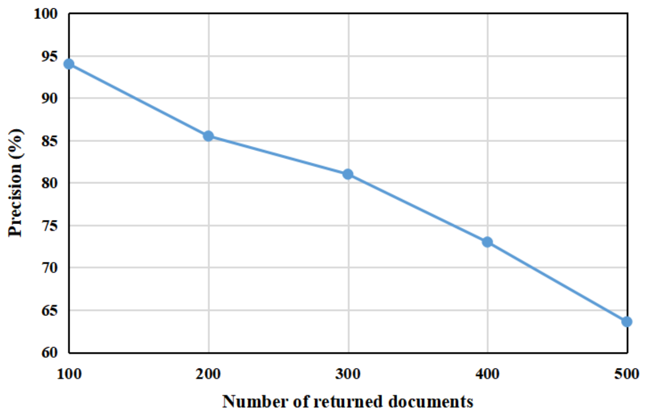

FIGURE 8. Precision.

Fig. 9 shows the recall rate of our scheme. According to the experiments, the number of all relevant documents in the documents collection is about 473. When the number of returned documents is 100, 200, 300, 400, and 500, the number of relevant documents in the search results is 94, 171, 243, 292, and 318, the recall rate is 19.8%, 36.1%, 51.3%, 61.7%, 67.2%, respectively. As the number of returned documents increases, the recall rate is on the rise. Because the recall rate represents the proportion of the number of relevant documents in the search results to all relevant documents in the documents collection, when the number of returned documents is small, the proportion of relevant documents in the search results to all relevant documents in the document collection is low, resulting in a low recall rate. When the number of documents

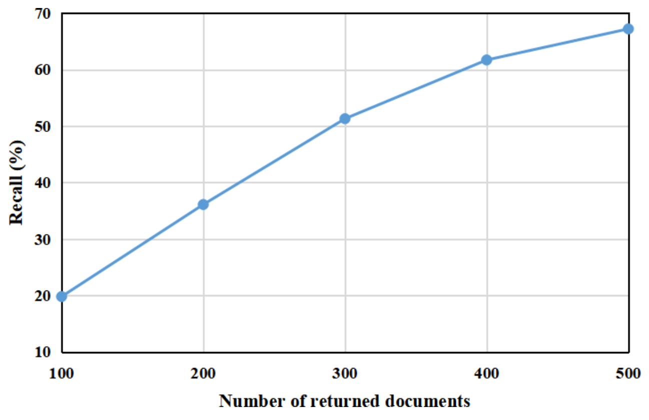

FIGURE 9. Recall.

in the search results is 100, the recall rate is about 19.8%. With the increase in the number of returned documents, more and more relevant documents are gradually searched, and the recall rate of the scheme will gradually increase. When the number of returned documents reaches 500, the recall rate of the scheme reaches 67.2%.

The keyword that users want to search may be mismatched to some form approximation keywords, but actually, they are not the keyword that users really want to query. For example, the keyword ''compute'' inputted by the user matches the keywords such as ''complex'' or ''compact'', but these two words are not consistent with the query intention of the users obviously. As our scheme is a fuzzy keyword searchable encryption scheme, matching these words with a similar form exactly reflects the fuzzy-search function of our scheme.

VI. CONCLUSION

In this paper, a dynamic and verifiable fuzzy keyword searchable encryption scheme based on blockchain is proposed. It solves the problem that the cloud server is semi-trusted and the blockchain-based searchable encryption only supports accurate keyword query. Combining blockchain technology with RSA dynamic accumulator, it allows the data owner to update document dynamically, and uses a smart contract to verify search results. The proposed scheme achieves service-payment fairness between the user and the cloud server, which enables the server and the user to search honestly and get correct retrieval results. In order to achieve the fuzzy keyword search, it generates the fuzzy keywords set by using edit distance. The cloud server can directly decrypt the corresponding encrypted index through searched fuzzy keyword, which simplifies the search process. Security analysis shows that the scheme can effectively achieve privacy protection and ciphertext retrieval. Performance analysis and experimental results show that the scheme is feasible in the process of fuzzy keyword searchable encryption. In a word, the scheme has functions such as update files dynamically, search results verification, fuzzy keyword search, and fair payment. In the future, we will design an effective symmetric searchable encryption scheme by obfuscating (document identifier, search token) inclusion/exclusion relationship and hiding the frequency of the search token to resist passive attack.

REFERENCES

[1] D. Xiaoding Song, D. Wagner, and A. Perrig, ''Practical techniques for searches on encrypted data,'' in *Proc. IEEE Symp. Secur. Privacy (S&P)*, Berkeley, CA, USA, May 2000, pp. 44–55, doi: 10.1109/SECPRI.2000.848445.

[2] E.-J. Goh, ''Secure indexes,'' Cryptol. ePrint Arch., Tech. Rep. 2003/216, 2003. [Online]. Available: https://eprint.iacr.org/index.html

[3] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, ''Searchable symmetric encryption: Improved definitions and efficient constructions,'' *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Nov. 2011, doi: 10.3233/JCS-2011-0426.

[4] S. Kamara, C. Papamanthou, and T. Roeder, ''Dynamic searchable symmetric encryption,'' in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, New York, NY, USA, 2012, pp. 965–976, doi: 10.1145/2382196.2382298.

[5] S. Kamara and C. Papamanthou, ''Parallel and dynamic searchable symmetric encryption,'' in *Proc. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2013, pp. 258–274, doi: 10.1007/978-3-642-39884-1_22.

[6] C. Guo, R. Zhuang, C.-C. Chang, and Q. Yuan, ''Dynamic multi-keyword ranked search based on Bloom filter over encrypted cloud data,'' *IEEE Access*, vol. 7, pp. 35826–35837, 2019, doi: 10.1109/ACCESS.2019.2904763.

[7] Q. Chai and G. Gong, ''Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers,'' in *Proc. IEEE Int. Conf. Commun. (ICC)*, Ottawa, ON, Canada, Jun. 2012, pp. 917–922, doi: 10.1109/ICC.2012.6364125.

[8] K. Kurosawa and Y. Ohtaki, ''UC-secure searchable symmetric encryption,'' in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, 2012, pp. 285–298, doi: 10.1007/978-3-642-32946-3_21.

[9] X. Jiang, J. Yu, J. Yan, and R. Hao, ''Enabling efficient and verifiable multi-keyword ranked search over encrypted cloud data,'' *Inf. Sci.*, vols. 403–404, pp. 22–41, Sep. 2017, doi: 10.1016/j.ins.2017.03.037.

[10] D. N. Wu, Q. Q. Gan, and X. M. Wang, ''Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting,'' *IEEE Access*, vol. 6, pp. 42445–42453, 2018, doi: 10.1109/ACCESS.2018.2861424.

[11] K. Kurosawa and Y. Ohtaki, ''How to update documents verifiably in searchable symmetric encryption,'' in *Proc. Int. Conf. Cryptogr. Netw. Secur.* Cham, Switzerland: Springer, 2013, pp. 309–328, doi: 10.1007/978-3-319-02937-5_17.

[12] L. Sardar and S. Ruj, ''FSPCDsse:A forward secure publicly verifiable dynamic SSE scheme,'' Cryptol. ePrint Arch., Tech. Rep. 2019/1123, 2019. [Online]. Available: https://www.iacr.org/

[13] R. Ramasamy, S. S. Vivek, P. George, and B. S. R. Kshatriya, ''Dynamic verifiable encrypted keyword search using bitmap index and homomorphic MAC,'' in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, New York, NY, USA, Jun. 2017, pp. 357–362, doi: 10.1109/CSCloud.2017.47.

[14] M. Islam, M. Kuzu, and M. Kantarcioglu, ''Access pattern disclosure on searchable encryption: Ramification, attack and mitigation,'' in *Proc. NDSS*, vol. 20, 2012. [Online]. Available: https://dblp.uni-trier.de/db/conf/ndss/ndss2012.html

[15] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, ''Leakage-abuse attacks against searchable encryption,'' in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2015, pp. 668–679, doi: 10.1145/2810103.2813700.

[16] C. Liu, L. Zhu, M. Wang, and Y.-A. Tan, ''Search pattern leakage in searchable encryption: Attacks and new construction,'' *Inf. Sci.*, vol. 265, pp. 176–188, May 2014, doi: 10.1016/j.ins.2013.11.021.

[17] H. Liu, B. Wang, N. Niu, S. Wilson, and X. Wei, ''Vaccine:: Obfuscating access pattern against file-injection attacks,'' in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Jun. 2019, pp. 1–9, doi: 10.1109/CNS.2019.8802803.

[18] G. Wang, C. Liu, Y. Dong, K.-K.-R. Choo, P. Han, H. Pan, and B. Fang, ''Leakage models and inference attacks on searchable encryption for cyber-physical social systems,'' *IEEE Access*, vol. 6, pp. 21828–21839, 2018, doi: 10.1109/ACCESS.2018.2800684.

[19] M. Giraud, A. Anzala-Yamajako, O. Bernard, and P. Lafourcade, "Practical passive leakage-abuse attacks against symmetric searchable encryption," in *Proc. 14th Int. Joint Conf. e-Bus. Telecommun.*, 2017, pp. 200–211, doi: 10.5220/0006461202000211.

[20] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. USENIX Secur. Symp.*, 2016, pp. 707–720.

[21] J. Ning, J. Xu, K. Liang, F. Zhang, and E.-C. Chang, "Passive attacks against searchable encryption," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 789–802, Mar. 2019, doi: 10.1109/TIFS.2018.2866321.

[22] C. Cai, J. Weng, X. Yuan, and C. Wang, "Enabling reliable keyword search in encrypted decentralized storage with fairness," *IEEE Trans. Depend. Sec. Comput.*, early access, Oct. 22, 2018, doi: 10.1109/TDSC.2018.2877332.

[23] H. Li, F. Zhang, J. He, and H. Tian, "A searchable symmetric encryption scheme using BlockChain," 2017, *arXiv:1711.01030*. [Online]. Available: http://arxiv.org/abs/1711.01030

[24] C. Cai, X. Yuan, and C. Wang, "Towards trustworthy and private keyword search in encrypted decentralized storage," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–7, doi: 10.1109/ICC.2017.7996810.

[25] H. G. Do and W. K. Ng, "Blockchain-based system for secure data storage with private keyword search," in *Proc. IEEE World Congr. Services (SERVICES)*, Honolulu, HI, USA, Jun. 2017, pp. 90–93, doi: 10.1109/SERVICES.2017.23.

[26] C. Cai, X. Yuan, and C. Wang, "Hardening distributed and encrypted keyword search via blockchain," in *Proc. IEEE Symp. Privacy-Aware Comput. (PAC)*, Washington, DC, USA, Aug. 2017, pp. 1–8, doi: 10.1109/PAC.2017.36.

[27] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 792–800, doi: 10.1109/INFOCOM.2018.8485890.

[28] Y. Zhang, R. Deng, X. Liu, and D. Zhang, "Outsourcing service fair payment based on blockchain and its applications in cloud computing," *IEEE Trans. Serv. Comput.*, early access, Aug. 7, 2018, doi: 10.1109/TSC.2018.2864191.

[29] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437–38450, 2018, doi: 10.1109/access.2018.2851611.

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

[31] Y. Zhang, R. H. Deng, J. Shu, K. Yang, and D. Zheng, "TKSE: Trustworthy keyword search over encrypted data with two-side verifiability via blockchain," *IEEE Access*, vol. 6, pp. 31077–31087, 2018, doi: 10.1109/ACCESS.2018.2844400.

[32] S. Wang, D. Zhang, and Y. Zhang, "Blockchain-based personal health records sharing scheme with data integrity verifiable," *IEEE Access*, vol. 7, pp. 102887–102901, 2019, doi: 10.1109/ACCESS.2019.2931531.

[33] L. Chen, W.-K. Lee, C.-C. Chang, K.-K.-R. Choo, and N. Zhang, "Blockchain based searchable encryption for electronic health record sharing," *Future Gener. Comput. Syst.*, vol. 95, pp. 420–429, Jun. 2019, doi: 10.1016/j.future.2019.01.018.

[34] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–5, doi: 10.1109/INFCOM.2010.5462196.

[35] C. Wang, K. Ren, S. Yu, and K. Mahendra Raje Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 451–459, doi: 10.1109/INFCOM.2012.6195784.

[36] C. Bösch, R. Brinkman, P. Hartel, and W. Jonker, "Conjunctive wildcard search over encrypted data," in *Proc. Secure Data Manage.* Berlin, Germany: Springer, 2011, pp. 114–127, doi: 10.1007/978-3-642-23556-6_8.

[37] X. Ge, J. Yu, C. Hu, H. Zhang, and R. Hao, "Enabling efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *IEEE Access*, vol. 6, pp. 45725–45739, 2018, doi: 10.1109/ACCESS.2018.2866031.

[38] S. Tahir, S. Ruj, A. Sajjad, and M. Rajarajan, "Fuzzy keywords enabled ranked searchable encryption scheme for a public cloud environment," *Comput. Commun.*, vol. 133, pp. 102–114, Jan. 2019, doi: 10.1016/j.comcom.2018.08.004.

[39] J. Wang, H. Ma, Q. Tang, J. Li, H. Zhu, S. Ma, and X. Chen, "Efficient verifiable fuzzy keyword search over encrypted data in cloud computing," *Comput. Sci. Inf. Syst.*, vol. 10, no. 2, pp. 667–684, 2013, doi: 10.2298/CSIS121104028W.

[40] X. Zhu, Q. Liu, and G. Wang, "A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China, Aug. 2016, pp. 845–851, doi: 10.1109/TrustCom.2016.0147.

[41] M. T. Goodrich, R. Tamassia, and J. Hasic, "An efficient dynamic and distributed RSA accumulator," 2009, *arXiv:0905.1307*. [Online]. Available: http://arxiv.org/abs/0905.1307

[42] Q. Xu, H. Shen, Y. Sang, and H. Tian, "Privacy-preserving ranked fuzzy keyword search over encrypted cloud data," in *Proc. Int. Conf. Parallel Distrib. Comput., Appl. Technol.*, Dec. 2013, pp. 239–245, doi: 10.1109/PDCAT.2013.44.

[43] F. Xhafa, J. Wang, X. Chen, J. K. Liu, J. Li, and P. Krause, "An efficient PHR service system supporting fuzzy keyword search and fine-grained access control," *Soft Comput.*, vol. 18, no. 9, pp. 1797–1802, 2013.

[44] Z. Liu, J. Weng, J. Li, J. Yang, C. Fu, and C. Jia, "Cloud-based electronic health record system supporting fuzzy keyword search," *Soft Comput.*, vol. 20, no. 8, pp. 3243–3255, Aug. 2016.

[45] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2706–2716, Dec. 2016.

[46] H. Zhong, Z. Li, J. Cui, Y. Sun, and L. Liu, "Efficient dynamic multi-keyword fuzzy search over encrypted cloud data," *J. Netw. Comput. Appl.*, vol. 149, Jan. 2020, Art. no. 102469, doi: 10.1016/j.jnca.2019.102469.

[47] G. Liu, G. Yang, S. Bai, Q. Zhou, and H. Dai, "FSSE: An effective fuzzy semantic searchable encryption scheme over encrypted cloud data," *IEEE Access*, vol. 8, pp. 71893–71906, 2020, doi: 10.1109/ACCESS.2020.2966367.

[48] J. Wang, X. Chen, H. Ma, Q. Tang, J. Li, and H. Zhu, "A verifiable fuzzy keyword search scheme over encrypted data," *J. Internet Serv. Inf. Secur.*, vol. 2, nos. 1–2, pp. 49–58, 2012.

[49] Y. Yang, Y.-C. Zhang, J. Liu, X.-M. Liu, F. Yuan, and S.-P. Zhong, "Chinese multi-keyword fuzzy rank search over encrypted cloud data based on locality-sensitive hashing," *J. Inf. Sci. Eng.*, vol. 35, no. 1, pp. 137–158, 2019.

[50] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Toronto, ON, Canada, Apr. 2014, pp. 1–9, doi: 10.1109/INFOCOM.2014.6848153.

[51] Y. Yang, S. Yang, and M. Ke, "Ranked fuzzy keyword search based on Simhash over encrypted cloud data," *J. Comput.*, vol. 40, no. 2, pp. 431–444, 2017.

[52] Y. Yang, J. Liu, S. Cai, and S. Yang, "Fast multi-keyword semantic ranked search in cloud computing," *J. Comput.*, vol. 41, no. 6, pp. 1346–1359, 2018.

[53] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Serv. Comput.*, vol. 13, no. 3, pp. 478–487, May 2020, doi: 10.1109/TSC.2017.2710190.

[54] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 715–725, Sep. 2017, doi: 10.1109/TSC.2016.2542813.

[55] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *Int. J. Commun. Syst.*, vol. 30, no. 1, p. e2942, Jan. 2017, doi: 10.1002/dac.2942.

[56] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multirecipient certificateless encryption with keyword search for cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2553–2562, Apr. 2020, doi: 10.1109/JIOT.2019.2943379.

[57] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Trans. Serv. Comput.*, early access, Apr. 11, 2019, doi: 10.1109/TSC.2019.2910113.

[58] K. Kurosawa, K. Sasaki, K. Ohta, and K. Yoneyama, "UC-secure dynamic searchable symmetric encryption scheme," in *Proc. Int. Workshop Secur.*, 2016, pp. 73–90, doi: 10.1007/978-3-319-44524-3_5.

XIXI YAN received the B.S. degree from the Huazhong University of Science and Technology, in 2006, the M.S. degree from Henan Polytechnic University, in 2009, and the Ph.D. degree from the Beijing University of Posts and Telecommunications, in 2012. She is currently an Associate Professor with the School of Computer Science and Technology, Henan Polytechnic University. She is mainly involved in digital content security and cloud computing.

XIAOHAN YUAN received the B.S. degree from Henan Polytechnic University, in 2018, where she is currently pursuing the M.S. degree. Her research interests include modern cryptography, networks, and information security.

QING YE received the B.S. degree from Northeastern University, in 2003, the M.S. degree from the Kunming University of Science and Technology, in 2008, and the Ph.D. degree from the Beijing University of Posts and Telecommunications, in 2014. She is currently a Lecturer with the School of Computer Science and Technology, Henan Polytechnic University. She is mainly involved in modern cryptography.

YONGLI TANG received the M.S. degree from Henan Polytechnic University, in 2005, and the Ph.D. degree from the Beijing University of Posts and Telecommunications, in 2008. He has visited Dublin City University, Ireland, in 2013. He is currently a Professor with the School of Computer Science and Technology, Henan Polytechnic University. His research interests include modern cryptography, networks, and information security.