Research Article

A Multiuser Ciphertext Search Scheme Based on Blockchain and SGX

Lianhai Wang, 1 Fengkai Liu, 1 Lingyun Meng, 1 Wei Shao, 1 Shujiang Xu, 1 Shuhui Zhang, 1 and Donghui Huang 2

1 Qilu University of Technology (Shandong Academy of Sciences), Shandong Computer Science Center (National Supercomputer Center in Jinan), Shandong Provincial Key Laboratory of Computer Networks, Jinan 250014, China
2 Rizhao Port, Group of Shandong Port, Rizhao 276800, China

Correspondence should be addressed to Lianhai Wang; wanglh@sdas.org

Received 31 July 2022; Accepted 6 September 2022; Published 23 September 2022

Academic Editor: Yinbin Miao

Copyright © 2022 Lianhai Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

To ensure the security of data, more and more users encrypt data for storage, which makes the high-efficiency ciphertext search problem in the context of cloud storage a research hotspot. Existing solutions still suffer from many vexatious problems, such as the need to maintain complex index structures and the unsatisfactory application of homomorphic schemes. To solve the above problems, this paper proposes a multiuser ciphertext search scheme based on blockchain and SGX. Our scheme uses blockchain and SGX to protect keywords and data privacy and complete decryption and keywords search of ciphertext data which does not need pregenerated indexes or preselected keywords. Second, for a multiuser scenario, a smart contract is designed to verify authorization requests and manage multiple authorized users. Finally, we give security analysis, function comparison, and performance analysis to prove the security and feasibility of our scheme. Experiments show that our scheme has effectively met practical requirements.

1. Introduction

With the rapid development of Internet of Things (IoT) technology, the amount of data generated by various applications has increased dramatically. To solve the problem of massive data storage, many enterprises and individuals choose to outsource data storage to cloud servers. Cloud storage can not only reduce local storage costs of users but also allow them to download and use the outsourced data regardless of device, access times, and geographical restrictions. However, it also brings problems such as data leakage and security risks. In 2021, data leakage incidents occurred frequently, which brought huge losses to enterprises and awakened society to the importance of data security. Therefore, privacy protection, data integrity, and sustainable services in the context of cloud storage have become major issues that cannot be ignored.

To prevent illegal data access by servers or unauthorized users, data should not be stored in plaintext. Users should encrypt data before uploading individual data to a cloud server. However, the commonly used data encryption scheme will limit the ability of the cloud server to process user access requests. In other words, the original search function of the cloud server will be invalid due to encryption, making the data retrieval a very difficult task. If a user wants to query data containing a certain keyword, he needs to download the encrypted data and then decrypt it for content search. The method is utterly inefficient since it requires extra space overhead and brings a poor experience to the user. Hence, the ciphertext search problem in the context of cloud storage has become a hot research topic in academia. In 2000, Song et al. [1] proposed a method that can perform a search on ciphertexts, namely, the symmetric searchable encryption (SSE) technology. At present, the research of SSE
technology for data retrieval has been developed to a certain extent. A typical SSE model is shown in Figure 1. A user first generates the index of a file from its keywords. A specific SSE algorithm is used to encrypt the file before it is uploaded to the cloud server. To fulfill a search, a data requester sends a Trapdoor to the cloud server. The cloud server searches the ciphertext for the Trapdoor and returns the search results. However, such models cannot guarantee data security when cloud servers are dishonest or compromised (centralization problems). To solve this problem, some schemes improved the security of data cloud storage by combining blockchain technology with SSE technology.

Blockchain [2] is a distributed database that creates a fully trusted environment between unfamiliar individuals without the need for third-party trust endorsements. Moreover, blockchain combined with cryptography technology can ensure transaction traceability, irreparable modification, and nonrepudiation. It is widely used for secure data sharing and large-scale collaborative computing and has been regarded as a powerful tool to solve data security and privacy issues in the context of cloud storage. At present, some schemes [3–10] combine blockchain and SSE technology. A user first stores encrypted data in Interplanetary File System (IPFS) and then sends the generated indexes and Trapdoor to smart contracts. Smart contracts can perform keyword search operations instead of cloud servers and finally return the search results to users, avoiding the centralization problem of cloud servers. However, the existing solutions mainly focus on keyword-based search. Users need to share keyword sets and encryption keys, which leads to many restrictions on the selection of keywords for users. The risk of data leakage by direct data sharing can also not be neglected. Since the search effect depends on the correlation between keywords and files, it is difficult to meet the data retrieval requirements for ciphertext in cloud storage systems. Furthermore, in the data search phase, a large amount of on-chain overhead can be costly for smart contracts to perform search operations. We need to consider moving data search operations to the off-chain to process.

Homomorphic encryption (HE) technology is an encryption method that allows direct operation on ciphertexts [11]. Some researchers use HE technology for ciphertext retrieval research [12–17]. Users first perform homomorphic encryption on files, under the premise of effectively protecting the privacy of users’ sensitive data, the cloud server is entrusted to directly perform addition and multiplication isomorphic operations on the ciphertext data, and the result is equivalent to the operation on the plaintext. However, such schemes suffer from a large computational overhead problem. Therefore, the homomorphic encryption scheme is difficult to apply in practice.

To solve the above problems, we propose a multiuser ciphertext search scheme based on blockchain and SGX. Specifically, to ensure the privacy of user data and solve the inability of smart contracts to be applied to complex computing scenarios, we combine blockchain and SGX to design a new ciphertext search model. The combination of blockchain and SGX will encounter challenges in data interaction and result traceability. But we solve the challenge of data interaction through on-chain contract storage and off-chain call acquisition. And then we record the search results on the blockchain to solve the traceability challenge of the search calculation result. In addition, for a multiuser scenario, we also design a smart contract to authorize and manage users.

In summary, the main contributions of this paper are shown as follows:

1. First, we propose a multiuser ciphertext search scheme based on blockchain and SGX. In this scheme, we use blockchain and SGX to protect keywords and data privacy and complete decryption and keywords search of ciphertext data which does not need pregenerated indexes or preselected keywords.

2. Second, for a multiuser scenario, a smart contract is designed to verify user authorization requests and manage multiple authorized users.

3. Finally, we give security analysis, function comparison with other schemes, and performance analysis to prove the security and feasibility of our scheme. Experiments show that our scheme has effectively met practical requirements.

1.1. Organization. The rest of this paper is organized as follows: Section 2 introduces related works. Section 3 presents preliminaries. Section 4 describes the system model, threat model, and design goals of the scheme. Section 5 presents the specific construction and protocol of the scheme. Section 6 delivers security analysis and functional comparisons. In Section 7, we test and analyze the functionality of the scheme. Finally, Section 8 concludes the paper.

2. Related Works

2.1. Searchable Encryption Schemes. At present, as a research hotspot in the field of cloud storage security, searchable encryption has made great progress. In 2000, Song et al. [1] proposed a symmetric searchable encryption scheme (SSE). SSE scheme [18, 19] has high encryption efficiency, but its
key management is more complicated in the data sharing phase. To solve this problem, Boneh et al. [20] proposed a public key searchable encryption scheme supporting keyword search, and to narrow down the keyword search, searchable encryption schemes [21–25] supporting multiple keywords were proposed. Xia et al. [26] proposed a multiplex-keyword sorting search scheme supporting dynamic updates. Li et al. [27] proposed a searchable encryption scheme using a fixed server to verify the user’s identity in an e-mail sending and receiving environment, which improves the security requirements of the scheme. Yang et al. [28] proposed a search scheme that supports both multiplex-keyword search and semantic sorting. Wang et al. [29] used each leaf node in a Merkle tree to store the MAC corresponding to the index and kept the root node as evidence locally. However, this scheme is implemented in a single-user model, which includes only two entities, the data owner and the server. Chen et al. [30] implemented forward and backward security in their scheme, but this scheme does not address the authorization problem well in a one-to-many model. The above schemes provide users with search results that satisfy the actual needs, but they fail to satisfy the needs of multiple users for data search and fail to achieve data access control. In addition, cloud encrypted data faces centralization problems such as server-side untrustworthiness and tampering of stored data.

In order to achieve multiuser access to data, in combination with attribute-based encryption (ABE), searchable encryption schemes can achieve keyword search while enabling fine-grained access control of encrypted files. Yin et al. [31] proposed a ciphertext policy attribute-based (CP-ABE) searchable encryption scheme, which has a high possibility of causing the server to return search results containing a large amount of irrelevant content and waste network bandwidth. Lin et al. [32] proposed an attribute set-based Boolean keyword search scheme, which can realize fine-grained access control and Boolean keyword search over encrypted personal health records (PHR). Zhang et al. [33] proposed a practical CP-ABE scheme, which offers users revocation and attribute update. The ciphertext size and decryption cost grow with the complexities of access policies. Mao et al. [34] gave the generic construction of Chosen-Plaintext Attack (CPA) secure CP-ABE scheme with verifiable outsourced decryption. Sun et al. [35] proposed a verifiable attribute-based ciphertext retrieval scheme, which allows multiple owners to encrypt and outsource their data to the cloud server independently. The scheme supports user attribute write-off and can verify the results returned by the server in a many-to-many scenario, but the scheme has large storage overheads. Miao et al. [36] proposed a secure multi-authority CP-ABKS (MABKS) system to avoid having performance bottleneck at a single point in cloud systems.

2.2. Searchable Encryption Schemes Based on Blockchain.

With the continuous maturity of blockchain technology, some schemes have introduced blockchain technology into searchable encryption to solve the centralization problems faced by encrypted data in the cloud such as server-side untrustworthiness and tampering of stored data. Zheng et al. [37] proposed a blockchain-enabled public key encryption scheme with multikeyword search (BPKEMS), which supports file updates. Moreover, a smart contract is used to ensure the fairness of transactions between the data owner and user without introducing a third party. Chen et al. [30] proposed a public key searchable encryption scheme in Vehicle Social Network (VSN), which replaces the original cloud server with a smart contract in the blockchain. Li and Wang et al. [38, 39] studied searchable encryption in a cloud environment, and two schemes reduced the search time under a large number of keywords. Jiang et al. [3] proposed a search scheme that supported multiple keywords and reduced the computation of the scheme and improved the efficiency of the scheme. Yang et al. [4] proposed a searchable encryption scheme in a shared electronic medical record scenario. The scheme stores the ciphertext of electronic medical records in the cloud server and the keyword ciphertext in the blockchain. Zhang et al. [5] introduced a dynamic accumulator algorithm into a blockchain searchable encryption scheme to improve the cryptographic search performance of the scheme. Guo et al. [6] designed a dynamic searchable encryption scheme based on the blockchain and used smart contracts in the blockchain to implement the verifiable function. Poongodi et al. [7] used the blockchain to design a trusted architecture using encryption and hashing methods to achieve reliable keyword search. Searchable encryption schemes generally suffer from high computational and storage overhead. Xu et al. [8] proposed a postquantum public key searchable encryption scheme on blockchain (PPSEB) for E-healthcare scenarios, which utilized a lattice-based cryptographic primitive to ensure the security of the search process and introduced blockchain technology to solve the problem of third-party untrustworthiness in the search process. Fu et al. [9] and Liu et al. [10] proposed a blockchain-based searchable encryption scheme in which the blockchain is used to store secure indexes and deploy smart contracts to perform the search of ciphertext files. All the above schemes are index-based ciphertext search schemes, which can effectively protect users’ data security because the search object is encrypted data. However, the semantic relationship of words is lost, and keyword search operation cannot be performed on ciphertext, so the index corresponding to encrypted data needs to be generated in advance, and a complex index structure needs to be maintained. In practical applications, users cannot search data beyond the predefined indexes and are restricted in the selection of keywords. In addition, a large amount of on-chain overhead is required in the data search phase, and a large amount of data needs to be considered to be processed off-chain.

2.3. Ciphertext Search Schemes Based on Homomorphic Encryption.

In recent years, in order to improve the accuracy and security of ciphertext retrieval, HE technology has attracted the attention of many researchers. The user first performs homomorphic encryption on plaintext data. On the premise of ensuring data privacy, the cloud server is
enthusiastic to perform homomorphic operation on ciphertext data directly, and the search result is equivalent to the operation on plaintext. In 1978, the HE scheme was first proposed by Rivest et al. [11]. In 2009, Gentry et al. [12] implemented homomorphic encryption theoretically. And then someone proposed the DGHV scheme [13] and the GSW13 scheme [14] in 2010 and 2013, where the former implemented an integer-based fully homomorphic encryption based on the approximate maximum convention number problem and the latter proposed the first identity-based homomorphic encryption scheme based on the error learning problem, to effectively guarantee the security of outsourced data and satisfy users to efficiently retrieve data stored in the cloud. In 2018, Fu et al. [15] proposed CRSHE: a new ciphertext retrieval scheme based on homomorphic encryption, which effectively solves the problems of privacy leakage of retrieved keywords and nonsupport of homomorphic encryption and improves the search efficiency and accuracy. In 2020, Han et al. [16] proposed a homomorphic encryption-based full-text retrieval scheme for cloud storage, which combines integer vector encryption technology with vector space model and is applied in full-text retrieval in third-party untrusted cloud storage. In 2021, Liu et al. [17] proposed a homomorphic encryption-based keyword search scheme in cloud servers, which has higher accuracy compared with the traditional ciphertext search scheme. Such schemes can directly perform computer on ciphertext data, but they require larger computational overhead or lower search efficiency, which is difficult to apply in practice.

3. Preliminaries

3.1. Blockchain Technology. The blockchain is a chained data structure formed by connecting multiple data blocks through a hash function, as shown in Figure 2. It realizes data verification, sharing, computing, storage, and other functions through a consensus mechanism. The blockchain provides a distributed trust ledger for each participant, and each node or user maintains and stores the same ledger to ensure that all users and nodes in the corresponding blockchain are completely consistent. A smart contract on a blockchain is an automatically running program that automatically performs some functions driven by time or events. It is a decentralized program code deployed in the blockchain to execute. Therefore, the smart contract provides programmability for the blockchain. The main high-level languages for writing smart contracts on Ethereum are Solidity, Serpent, and LLL; it is implemented by storing it compiled into bytecode.

3.2. SGX. The wide application of blockchain technology enables applications to ensure the security of data on the chain. However, in the off-chain processing of data, blockchain technology cannot guarantee its security, so off-chain data processing requires a trusted execution environment based on hardware or software.

The solution adopted in this paper is to rely on the software protection extension Software Guard Extensions (SGX) [40] launched by Intel. The SGX is an extension of the Intel instruction set architecture, which provides hardware-level security for the operation of the program, rather than based on external software. It allows the application to open up a protected and trusted executable area in the memory, called an Enclave. The Enclave provides integrity protection for the programs. If someone attempts to access the Enclave outside the safe area, he will be rejected. After the data in the Enclave is transmitted to the nonsafe area through special encryption, even if other machines get the encrypted data, encrypted data cannot be decrypted, which ensures the correctness and confidentiality of the data.

4. Problem Formulation

In this section, we describe in detail the system model, threat model, and design goals in the scheme and design a multi-user ciphertext search scheme based on blockchain and SGX.

4.1. System Model. In this scheme, the proposed system model mainly contains five entities: data owner A, data requester B, data storage DS, blockchain BC, and query node S, as illustrated in Figure 3.

4.1.1. Data Owner A. The data owner is an entity with a large amount of data but limited resources. He uses a key to encrypt personal data and upload it to the data storage. And data authorization requests and authorized users are verified and managed; different data owners allow different query nodes to perform data searches on the ciphertext.

4.1.2. Data Requester B. The data requester is the entity that requests the ciphertext data to search. When the data requester wants to search for keywords, he needs to send a data authorization request to data owner A and finally obtains the plaintext data after decryption according to the encrypted search result.

4.1.3. Data Storage DS. Data storage is a kind of platform that provides distributed storage service for data owner A. It has huge storage space and powerful computing power; however, it is not trustworthy. When uploading ciphertext data, it first verifies the integrity of the data, stores the related data after successful verification, and returns the corresponding storage hash address.
4.1.4. Blockchain BC. It is a public chain composed of data owners, data requesters, and query nodes. Anyone can join this public blockchain to view or publish transactions. It is mainly used for data storage, transaction recording, and smart contract deployment.

4.1.5. Query Node S. The query node is a registered node on the blockchain. It has its own corresponding SGX trusted execution environment and can take advantage of its own SGX trusted hardware. SGX is a trusted and independent execution environment that exists independently of an untrusted operating system, providing a safe and confidential space for private data and sensitive computing in an untrusted environment.

4.2. Threat Model. In this scheme, data request B is considered semicredited. Only after the verification of the authorization request, data request B can obtain the search permissions for encrypted data. We assume that the key storage of data owners A and data requester B is safe, not attacked by attackers, and all search tasks are performed in SGX Enclave. Secondly, external attackers steal the transmitted data transmitted through public channels and hope to read or modify the data of data owner A. In addition, if the storage data is not accessed for a long time, it may cause loss and other conditions.

4.3. Design Goals. In this scheme, our main design goals are as follows:

(1) Data privacy: In this scheme, since the personal data of data owner A is very sensitive, data security protection is necessary, so in this scheme, the data is encrypted and uploaded to the data storage DS to store.

(2) Storage integrity: In this scheme, the data storage DS stores the encrypted data only when the integrity of the data is verified.

(3) User access control: In this solution, when data requester B wants to search the personal data of data owner A, he needs to send an authorization request to data requester A, and the search permission can only be obtained after the authorization is successful.

(4) Privacy of keywords: In this scheme, due to the encrypted transmission of query keywords, other entities cannot obtain any information about keywords through the ciphertext of the keyword.

5. Our Scheme

In this subsection, we introduce the system construction and protocol of this scheme in detail. This scheme includes six phases: system initialization phase, data processing phase, data storage phase, user authorization phase, data search phase, and data decryption phase. And Table 1 gives some important notations and descriptions used in the following paper. Details are as follows.

5.1. Scheme Construction

5.1.1. System Initialization Phase. Based on the security parameters, the keys and system parameters are generated in the following ways, as follows.

We choose the secure SHA-256 hash algorithm hash \((m)\) and RSA signature algorithm \(\text{Sign}_{SK}(m)\), where \(\text{Sign}_{SK}(m)\) denotes that SK signs \(m\). The system calls the ECC algorithm to generate the key pair \((PK_o, SK_o)\) and \((PK_p, SK_p)\) for data owner A and data requester B, respectively, and then calculates their Ethereum network addresses addrA and addrB.

The data owner A uses the DES algorithm to generate a
symmetric key k, generating the Enclave key pair (PK_{sgx}, SK_{sgx}) of the SGX corresponding to the query node. System parameters is para = \{\text{hash}(m), \text{Sign}_{SK_{sgx}}(m), \text{addr} a, \text{addr} b\}.

5.1.2. Data Preprocessing Phase. As a registered user of the blockchain, data owner A first divides M into equal-sized data \(m_i\) \((i = 1, 2, \ldots, n)\) and uses a symmetric key k to encrypt a plaintext data collection \(M = \{m_1, m_2, \ldots, m_n\}\) and generate a ciphertext data collection \(C = \{c_1, c_2, \ldots, c_n\}\), where the ciphertext data \(c_i = \text{Enc}_{k}(m_i)\), and then generate a ciphertext data hash collection \(H = \{h_1, h_2, \ldots, h_n\}\), where the ciphertext data hash value \(h_i = \text{hash}(c_i)\), and finally write search program P.

5.1.3. Data Storage Phase. Data owner A uses his own private key SK_a to sign PK_a and H and P to generate \(r = \text{sign}_{SK_a} (PK_a[H||P])\) and send the outsourced collection Set = \(\{PK_a[H||P], r\}\) to the data storage DS. After the data storage DS receives the Set, first, it verifies the validity of the signature r and the integrity of the ciphertext data collection C. If the verification passes, store \(H||C||P\) in the data storage DS, and then data storage DS returns the corresponding storage hash address \(\text{addr} = [\text{addr} 1, \text{addr} 2, \ldots, \text{addr} n]\). Otherwise, the verification fails, and the outsourced collection set needs to be reuploaded.

The data owner A uses the SGX remote authentication mechanism to authenticate the identity information of the SGX Enclave of the query node. After the authentication is passed, the query node obtains the search program P from the data storage DS and then installs and deploys it in the Enclave. The data owner A uses the PK_{sgx} encryption key k, organizes the hash value collection H, and stores the hash address addr and other information, generates a timestamp ts1, and finally records the txdata in the blockchain. Algorithm 1 describes the process of data owner A’s data transaction txdata generation.

5.1.4. User Authorization Phase. If data requester B wants to search the ciphertext data of data owner A, he first uses private key SK_b to encrypt his public key PK_b and query keyword \(w\) CT = \text{Enc}_{SK_b}(w) and then sends the data authorization request CT in the form of a transaction to data owner A. After the data owner A receives the data authorization request CT, he first uses addr_b to query whether the legal user list in the smart contract contains this user. If the query is successful, it means that the data requester B can obtain the search permission; if the query fails, the data owner A then decrypts the CT using the data requester B’s public key PK_a and then performs the Keccak-256 hash operation on the public key PK_a of the data requester B, truncates the last 20 bytes into the string R1, and finally calls the smart contract to R1 matches addr_b. If it returns 1, it means that the authorization request is verified, and the data owner A adds the data requester B to the list of legal users through the smart contract; if it returns 0, it means that the data requester B cannot obtain the search permission. In addition, data requester B can also be removed or revoked from the list of legal users through the smart contract.

After the verification is passed, the data owner A uses k to encrypt w to generate a Trapdoor \(Tr = \text{Enc}_{k}(w)\) and sends Tr to the query node through the blockchain.

5.1.5. Data Search Phase. The Intel SGX extension employs two data sealing schemes: the safe zone strategy (MRENCLAVE) policy and the sealed strategy (MRSIGNER) policy. In this scheme, MRSIGNER is used to query nodes and generate public and private key pairs (PK_{sgx}, SK_{sgx}) in the Enclave, the private key obtains key (SK_{sgx}) through the MRSIGNER policy and outputs it to the nonsecure area, and the public key is explicitly output to the nonsecure area and uploaded to the chain. The query node obtains the Tr sent by the data owner A through the blockchain and performs data decryption and keyword search operation in SGX trusted execution environment. The query node sends the final search results to the blockchain.

Specifically, the SGX trusted execution environment corresponding to the query node first performs integrity verification on the ciphertext data and executes step (1). The SGX trusted execution environment corresponding to the query node performs decryption and keyword search operations and executes steps (2)–(6). The query node sends the encrypted search result to the data requester B and executes step (7), taking the search for ciphertext data \(c_i\) as an example. Algorithm 2 describes the process of decryption and keyword search:

1. The query node first uses the MRSIGNER policy of the corresponding SGX, decrypts key (SK_{sgx}) to obtain SK_{sgx}, then obtains the ciphertext data \(c_i\) and its corresponding hash value \(h_i\) through the smart contract, regenerates the ciphertext hash value \(h\) and calculates whether \(h_1\) and \(h\) are equal to verify data integrity.
2. The SGX corresponding to the query node uses the private key SK_{sgx} to decrypt Enc_{PK_{sgx}}(k) to obtain the key \(k = \text{Dec}_{SK_{sgx}}(\text{Enc}_{PK_{sgx}}(k))\).
3. The SGX corresponding to the query node uses the key \(k\) to decrypt the ciphertext data \(c_i\) to obtain the plaintext data \(m_i\) for performing the search task.
5.1.6. Data Decryption Phase. Data requester B obtains $E_{data}$ from the blockchain and decrypts $E_{data}$ using the private key $SK_a$ to obtain the corresponding plaintext data.

5.2. Our Protocol. In this scheme, the protocol flow includes the following 11 steps, where Step 1 describes the data preprocessing phase, Steps 2–3 describe the data storage phase, Steps 4–5 describe the user authorization phase, Steps 6–8 describe the data search phase, and Step 9 describes the data decryption phase. The logical process is shown in Figure 4.

Step 1. The data owner A encrypts data with an encryption key $k$ and uploads it to the data storage DS.

Step 2. Data storage DS receives the encrypted data. If the verification passes, it returns the corresponding hash address; otherwise Step 1 needs to be performed again.

Step 3. Data owner A records hash address and encrypted key on the blockchain in the form of transactions for data sharing.

Step 4. If data requester B wants to search the encrypted data of data owner A, data requester B needs to send an authorization request to data owner A.

Step 5. Data owner A performs authorization verification and updates the legal user list after the verification is passed and uploads the Trapdoor to the blockchain.

Step 6. The query node obtains the Trapdoor, hash address, and encrypted key from the blockchain and then downloads encrypted data from the data storage DS.

Step 7. The query node performs decryption and keyword search operations of ciphertext in the SGX-based TEE.

Step 8. The query node sends the encrypted search results to the blockchain.

Step 9. Data requester B obtains the encrypted search results from the blockchain and then decrypts the search results.

Algorithm 1: Privacy protection of data owner A’s data.

```
Input: Session key $k$; A’s data $M = \{m_1, m_2, \ldots, m_n\}$; Search $P$
Output: Transaction $tx$
(1) Encrypted Owner data $M = \{m_1, m_2, \ldots, m_n\} c_i = Enc_k(m_i)$ and $C = \{c_1, c_2, \ldots, c_n\}$
(2) Encrypted $k$ key = $Enc_{PK_a}(k)$
(3) Set $h_i = \text{hash}(c_i)$ and $H = \{h_1, h_2, \ldots, h_n\}$
(4) Set $r = \text{sign}_{SK_a}(PK_a || H \| P)$
(5) Set $Set = (PK_a || C || H \| P || r)$
(6) Send $Set$ to the DS and get $addr$
(7) Generate timestamp $ts1$
(8) Set $tx = [Ekey, H, addr, ts1]$
(9) Return $tx$
```

Algorithm 2: Data acquisition and search computation.

```
Input: Enc_k(w); Enc_{PK_a}(k); c_i; h_i; count;
Output: search result;
(1) Set $h = \text{hash}(c_i)$
(2) if $(h = h_i)$ then
(3) Decrypt $Enc_{PK_a}(k)k = Dec_{SK_a}(Enc_{PK_a}(k))$
(4) Decrypt $Enc_k(w)w = Dec_k(Enc_k(w))$
(5) Decrypt $c_i = Dec_k(c_i)$
(6) Search computation with $m_i$
(7) if (count != 0) then
(8) Encrypted $m_i E_{data} = Enc_{PK_a}(m_i)$
(9) return $E_{data}$;
(10) else
(11) return false;
```
6. Security and Function Analysis

6.1. Security Analysis. Since the personal data of the data owner is sensitive and private, the security of the data is of great importance in this scheme. Data security is analyzed in the following aspects.

6.1.1. Data Security. In this solution, to protect data security, the personal data of data owner A is encrypted by the key \( k \) and stored in the data storage DS, and data requester B can obtain the search permission only through an authorization request. First, the data owner A uses the key to encrypt the data and store it in the data storage DS. Anyone can find the encrypted data through the storage hash address in the blockchain. To decrypt the encrypted data, the attacker must obtain the key \( k \). However, the key \( k \) is only stored locally in data owner A and SGX security zone. Assuming that data owner A does not leak the private key, the attacker cannot obtain the key \( k \). Therefore, the security of the personal data of data owner A is guaranteed.

6.1.2. Signature Forgery. In this scheme, the correct storage of data is guaranteed through the basic principle of signature. The user signs the ciphertext data and uploads it to the data storage DS. When the private key of the data owner is securely stored, the attacker cannot forge the signature, so other entities cannot destroy the authenticity of the data upload by forging the signature.

6.1.3. Tamper-Proof. In the data upload storage phase, there may be malicious users tampering with the blockchain information or transaction information. In this plan, the blockchain setup is POA consensus. Each block is generated by the certification node. For the compulsory process to verify the identity, the right to generate new blocks can be obtained. Malicious nodes cannot know the private key of the credible certification nodes, and it is impossible to fake the identity of the certification node to pack the block or modify the signature of the block information. Malicious nodes are difficult to tamper with data on the blockchain and data stored on the blockchain to ensure the authenticity and accuracy of the data.

6.1.4. Data Privacy. In this solution, to ensure the privacy of the data, data owner A uses a symmetric encryption algorithm to protect personal data, and data requester B can obtain the search permission only after the authorization is successful. In addition, in the data search stage, the encrypted personal data and query keywords are read in the nonsecure area of SGX, encrypted data can only be decrypted in the secure area of SGX, and the decrypted data cannot be obtained in the nonsecure area of SGX. Finally,
use the public key of data requester B to encrypt the search results in the secure area of SGX and output them in the nonsecure area of SGX. Even if other machines steal the encrypted data in the nonsecure area, the data cannot be decrypted on the personal machine. Thus, the privacy and security of the data in the data upload stage and the data search phase are guaranteed.

6.2. Function Analysis. In Table 2, we compare our scheme with existing schemes in terms of functionality and security. We can see that schemes [10, 15, 29, 30] require pregenerated indexes and do not better solve the authorized access problem. Schemes [10, 30] use smart contract to replace cloud servers to perform search tasks, which can solve the centralization problem of cloud servers in schemes [15, 29], but schemes [10, 30] still have a high risk of data leakage and are difficult to apply to complex computing scenarios problems. Also, schemes [29, 30] are suitable for single-user scenarios. However, this scheme does not require pregenerated indexes and is suitable for multiuser scenarios. Our scheme also uses TEE to better protect user data security and perform decryption and full-text search under the SGX off-chain to solve the problem that smart contracts cannot be applied to complex computing scenarios.

7. Experiment and Analysis

In this subsection, the practicality and feasibility of this scheme will be tested and analyzed through experiments. We installed Ubuntu 20.0 on a computer with Intel(R) Core(TM) i7-9750H CPU@2.60 GHz, 16 GB RAM, Microsoft Windows 10 operating system, and then performed simulation experiments. We use the DES symmetric encryption algorithm to encrypt the data, the blockchain part uses the Ethereum private network built by Geth, the smart contract is written in solidity language, and the search program P is written in C++. Next, we mainly tested and analyzed the implementation and gas costs of the contract and the performance of this scheme.

7.1. Implementation and Gas Costs. To discuss the feasibility of smart contract in this scheme, we implemented it on Rokeby (an Ethereum test network) where Rinkeby not only provides free funding requests but also designs a user interface for a convenient block resource manager. In addition, we employed a Google Chrome plugin (MetaMask-Chrome) to link Rinkeby in Chrome and use Remix10 to deploy and invoke smart contracts; the details of this implementation are shown below.

(1) First, we used MetaMask to generate two users (data owner A and data requester B) for our test with addresses 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 and 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2, then switched to A’s account, and requested 3 Ether from Rinkeby so that A could deploy contracts and generate data transactions, etc.

(2) Then, we simulate user A. We use Rinkeby to deploy the smart contract to the blockchain and get its address (0x9D7f74d0C41E726EC95884E0e97-Fa6129e3b5E99), and we also call the autuser algorithm via Remix to verify that B’s authorization is passed or not.

(3) Next, we simulated A updating and viewing the list of legitimate users. Update the authorized user B to Rinkeby by calling the adduser algorithm, and view the list of legal users by calling the getuser algorithm. Here getuser is designed as a view type algorithm that will not modify the state of the smart contract (therefore, there is no transaction confirmation time).

(4) Finally, we also simulated that A deletes the authorized user B from the list of legitimate users. The authorized user B is removed from the legal user list by calling the deluser algorithm, where the algorithm can only be called by A.

Additionally, to test the cost in terms of transaction fees, we evaluated the gas cost of these operations (i.e., authorize, adduser, getuser, deluser). As it can be seen from Table 3, the biggest cost is to deploy a smart contract, which is about 5.0881 USD, but it only needs to be executed once. Although other operations are called repeatedly, their cost is about 1 USD (especially the cost of getuser is about 0.1709 USD), which means that A only needs to spend 0.1709 USD to view the list of legitimate users, which is an acceptable cost even if it is called repeatedly.

7.2. Performance Test and Analysis

7.2.1. Cost of Cryptographic Primitives. We use symmetric encryption algorithm to perform user encryption and SGX decryption tests on file data, respectively, and compare the
average calculation time of different sizes of data. Each experiment is repeated 1000 times and the average time is calculated. The test results are shown in Figures 5 and 6. As can be seen from the figures, the average time of user encryption and SGX decryption increases gradually with the increase of data volume.

7.2.2. Search Performance. To verify the performance of this scheme in terms of search efficiency, as shown in Figure 7, we use the control variable method and the length of the keyword as 2 characters and test the average time of different numbers of keywords. The unit of time is milliseconds (ms), each experiment is repeated 1000 times, and the average time is calculated. From the curve shown in Figure 7, it can be seen that, with the increase in the number of keywords, the search time increases; this scheme only designs a simple keyword matching operation in the data search phase, which is related to the number of keywords. For encrypted data stored in the cloud environment, the search time efficiency of this scheme is reasonable.

8. Conclusion

In this paper, we propose a multiuser ciphertext search scheme which uses blockchain and SGX to protect keywords and data privacy and performs the decryption and keywords matching of ciphertext data in SGX. To fit searching in multiuser scenarios, we design a smart contract to realize user authorization and management. The security analysis, function comparison, and performance analysis prove that our scheme meets the security and privacy requirements. In the future, we will carry out further research work on multikeyword ciphertext search based on blockchain and SGX in specific scenarios.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Shandong Provincial Key Research and Development Program (2021CXGCO10107 and 2020CXGCO10107) and the National Natural Science Foundation of China (62102209).
References


