TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/18 pp13–26
DOI: 10.26599/TST.2021.9010070
Volume 28, Number 1, February 2023

Privacy-Preserving Searchable Encryption Scheme Based on Public and Private Blockchains

Ruizhong Du, Caixia Ma*, and Mingyue Li

Abstract: While users enjoy the convenience of data outsourcing in the cloud, they also face the risks of data modification and private information leakage. Searchable encryption technology can perform keyword searches over encrypted data while protecting their privacy and guaranteeing the integrity of the data by verifying the search results. However, some associated problems are still encountered, such as the low efficiency of verification and uncontrollable query results. Accordingly, this paper proposes a Privacy-Preserving Searchable Encryption (PPSE) scheme based on public and private blockchains. First, we store an encrypted index in a private blockchain while outsourcing corresponding encrypted documents to a public blockchain. The encrypted documents are located through the encrypted index. This method can reduce the storage overhead on the blockchains, and improve the efficiency of transaction execution and the security of stored data. Moreover, we adopt a smart contract to introduce a secondary verification access control mechanism and restrict data users' access to the private blockchain through authorization for the purpose of guaranteeing data privacy and the correctness of access control verification. Finally, the security analysis and experimental results indicate that compared with existing schemes, the proposed scheme can not only improve the security of encrypted data but also guarantee the efficiency of the query.

Key words: private blockchain; public blockchain; access control; forward privacy; backward privacy

1 Introduction

As the demand for storage and computing resources continues to grow, organizations demonstrate a strong trend of outsourcing data to third parties, such as cloud servers. Because outsourced data contain sensitive information, Data Owners (DOs) usually choose to encrypt their data before outsourcing to the cloud. Therefore, searching for encrypted data has become a major problem under the premise of guaranteeing data privacy. In response to this problem, Song et al.[1] first proposed Searchable Symmetric Encryption (SSE). This research subsequently initiated a series of related studies.

Searchable Encryption (SE) mainly focuses on the function of retrieving data that cannot be solved by traditional encryption technology. Generally, SE schemes work by generating an encrypted index. The DO outsources the index together with the encrypted data to the service provider. The Data User (DU) delivers the search token for a specific keyword, and the service provider identifies matches by executing search algorithms using the token and encrypted index. Early schemes returned results via scanning, and the efficiency linearly decreased with the increase in the amount of data in the database. To solve the efficiency problem, many researchers have improved SE technology by building an index and extracting keywords so that the query complexity is only related to the keywords in the file set. However, most of the early schemes are static and cannot be updated dynamically.

When storing data on a cloud server, users usually

• Ruizhong Du and Caixia Ma are with School of Cyberspace Security and Computer, and Hebei Province Key Laboratory of High Confidence Information System, Hebei University, Baoding 071002, China. E-mail: drzh@hbu.edu.cn; macaixia20@163.com.
• Mingyue Li is with the College of Cyber Science, Nankai University, Tianjin 300350, China. E-mail: limingyue@mail.nankai.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2021-09-15; accepted: 2021-09-29

need to update data. To meet these needs, dynamic SE technology has emerged, which makes the SE scheme more flexible and accessible. However, the attacker can also observe the process of data updating and find the links between keywords and files to steal or change the data, making the security analysis more complex. Therefore, in the security analysis of dynamic SE, two security attributes—forward privacy and backward privacy—are proposed. Forward privacy ensures that new files will not have any connection with previous search operations, whereas backward privacy guarantees that deleted documents will not leak any information due to subsequent search operations.

With the emergence of Bitcoin, researchers have begun to use blockchains to store data. Therefore, currently, blockchains are widely used in the field of SE[2, 3]. Because data on blockchains are open, unchangeable, and irrevocable and all operations are transparent and reliable, blockchains can effectively protect the integrity and privacy of data and prevent cloud storage information modification. To solve the data privacy, integrity and correctness problems, this paper proposes a Privacy-Preserving Searchable Encryption (PPSE) scheme with an access control mechanism based on private and public blockchains. The PPSE uses the blockchain to replace the central server and outsources the search query to a smart contract, which produces a correct and unchangeable result without verification of the DO. In addition, an access control mechanism is introduced through smart contracts to restrict DUs' access to the private blockchain, which can protect data privacy and eliminate malicious attacks. This research makes the following contributions to the fields of blockchains and SE:

(1) Encrypted indexes and documents are uploaded to private and public blockchains, respectively, and encrypted documents are located through encrypted indexes, which can reduce the storage overhead on blockchains and improve the efficiency of transaction execution and the security of stored data. At the same time, the PPSE can guarantee the integrity of query results without designing a special verification mechanism.

(2) To protect data privacy, prevent privacy leakage, and avoid network attacks, we introduce a secondary verification access control mechanism by adopting smart contracts. We embed it in the private blockchain and verify DUs twice to restrict their access to the private blockchain, which improves the precision of verification.

(3) Through a simulation experiment of the local test network, a large amount of data are tested and analyzed. The security analysis and experimental results indicate that compared with existing schemes used in the cloud, the PPSE can not only improve the security of the data but also guarantee the efficiency of queries. It can also better improve the efficiency of queries and achieve type II backward privacy compared with the scheme in the blockchain.

The remainder of this paper is organized as follows: The related work is presented in Section 2. The system model, along with the required knowledge, are discussed in Section 3. An overview of the architecture is presented in Section 4. The proposed PPSE framework is presented in Section 5. The security and formula analyses are discussed in Sections 6 and 7, respectively. The conclusions of the paper are summarized in Section 8.

2 Related Work

SSE is an encryption primitive that allows the server to directly search over encrypted data. It was first proposed by Song et al.[1]. Because the search process needs to scan collected data to return results, its efficiency will linearly decrease as the database grows. To improve the efficiency of queries, many scholars have conducted research on this aspect. However, most early schemes do not support dynamic updating operations[4].

With the changes in users' requirements, dynamic SE technology has been proposed[5]. Accordingly, a client can add or delete data at any time, which makes the solution flexible and practical, but it will cause data leakage. To solve this hidden problem of information leakage during the update process, Stefanov et al.[6] proposed the concept of forward and backward privacies in 2014. Forward privacy guarantees that new files added will not have any connection with previous search operations. In addition to the apparent benefits of the "dynamic" construction of encrypted datasets, it is also critical for other abuse attacks, such as those relying on hostile file injection. Backward privacy guarantees that deleted documents will not leak any information due to subsequent search operations. Subsequently, Bost[7] formally defined backward privacy. In 2017, they proposed an SE scheme that satisfies forward and backward privacies and divided the backward privacy level into three types from low to high: Type III, Type II, and Type I. To further examine backward privacy, a variety of schemes have been proposed[8, 9], some of which can achieve Type I

backward privacy. Subsequently, Sun et al.[10] proposed a symmetric puncturable encryption scheme in 2018, which makes the server lose the ability to search for deleted documents containing certain keywords and provides a new form of encryption for the field of SE. In 2021, Patranabis and Mukhopadhyay[11] proposed a dynamic SE scheme that satisfies forward and backward privacies and supports conjunctive keyword searches. In addition, to adapt to more advanced functions, more researchers have performed a series of works on other aspects, such as query function[12, 13] and performance optimization[14–16].

Most existing solutions mainly focus on an honest but curious cloud server, and security designs against a malicious server have not drawn enough attention. When an external attack or internal configuration error occurs, the cloud server becomes a malicious server, which then leads to changes or the disclosure of encrypted data and may return wrong query results. In response to the above problems, many verification schemes have been proposed. In 2019, Soleimanian and Khazaei[17] used a pseudorandom function and one-way function to complete the verification of open results. In 2020, Tong et al.[18] combined the Merkle hash tree and k-means clustering and proposed a scheme that improves the efficiency of verification and security. Yang and Zhu[19] solved the correlation verification problem in a semantic environment by transforming the verification process into a linear programming task. These schemes can obtain correct results when users are trusted, but correct data cannot be easily obtained when users are not truthful.

To solve the problem of having incorrect query results caused by untruthful users, blockchain technology has been introduced into SE. It can effectively guarantee that encrypted data will not be tampered with and ensure the integrity of query results. The blockchain-based solution proposed by Tang[20] can achieve the required fairness while retaining the privacy of the original SE scheme. Hu et al.[21] proposed an SSE scheme using smart contracts. The index of user files is stored in a smart contract in a peer-to-peer network, and these files can be stored in any public cloud storage system. However, the premise allowing for security to be guaranteed is that the blockchain is safe enough. Subsequently, Chen et al.[22] modified the index structure based on the scheme of Hu et al. and applied it to the electronic case sharing system. The experiment indicated its usability, but the overhead was large. Jiang et al.[23] designed a publicly verifiable search framework for outsourced encrypted data based on a blockchain and constructed a stealth authorization scheme to make the blockchain more secure. They uploaded the encrypted index on the blockchain while outsourcing the corresponding encrypted data to the cloud, which is a safe and feasible approach for dealing with index query results. However, similar to the scheme proposed by Guo et al.[24] to verify query results using a blockchain, encrypted data stored in the cloud server may be leaked or changed. As a result, data encounter potential safety hazards, and query results are uncontrollable.

3 Background

3.1 Blockchain

A blockchain is a distributed public database based on blockchain technology, which combines data exchange, processing, and storage formed among multiple participants based on modern cryptography, distributed consensus protocol, point-to-point network communication technology, and smart contract programming language[25]. Ethereum used in the PPSE is a new type of decentralized computing platform based on blockchains. Ethereum allows users to perform any complex operation as needed and implement specific operations by adopting smart contracts.

A blockchain has transparency, public verification, unchangeability, and unforgeability characteristics. Its most significant advantage is decentralization, allowing it to support data verification, sharing, computing, storage, and other functions through multilateral autonomous technical means, such as consensus[26]. The advantages of blockchains have attracted companies to develop blockchain-based applications. Based on users' access rights to a blockchain, it is divided into public and private blockchains.

3.1.1 Public blockchain

A public blockchain refers to a consensus blockchain in which any user can read, send transactions, and obtain valid confirmation. The security of a public blockchain is maintained by the workload or equity certification mechanism. These blockchains exist by combining economic rewards with encrypted digital verification and adhere to the following general principle: the economic reward everyone receives from blockchains is proportional to the contribution made to the consensus process. This incentive mechanism can also encourage more participants to join the blockchain network so that

the concept of blockchain projects can be spread to a wide audience. Therefore, for a public blockchain, an incentive mechanism is very important for project operations. These blockchains are generally considered "fully decentralized". A public blockchain has a network effect, high resistance to censorship, neutrality, openness, decentralization, irreversibility, and irrevocability.

3.1.2 Private blockchain

A completely private blockchain refers to a blockchain with writing permission only in the hands of one organization. Data access and writing have very strict permissions. In most cases, such data are not publicly readable. Essentially, compared with a system that is completely open and uncontrolled and guarantees network security through an encrypted economy, the private blockchain can create a system with stricter access control, and modification or even reading permissions can be limited to a few users. At the same time, this system still retains the authenticity and partial decentralization of a blockchain. The private blockchain has limited read permissions, low transaction costs, easy-to-modify rules, and few nodes participating in blockchain activities, which make various operations in the private blockchain highly efficient.

3.2 Smart contract

Smart contracts in Ethereum are applications with a state stored in the blockchain. They can facilitate, verify, and enforce the process of the contract[27]. Each smart contract, identified by a special address, consists of script code, currency balance, and storage space in the form of a key/value store. Once created and deployed to Ethereum, even its creator cannot modify the code of the contract forever[18].

3.3 Index structure

Because the search of the forward index takes too long, the PPSE uses a KC-IDC index structure (KC is the number of times the keyword appears in the document ID, and IDC is the number of documents containing the keyword). This index structure can quickly obtain the document list containing the keyword to improve query efficiency. It is a specific storage form that realizes "Keyword: Document Matrix", which is mainly composed of a keyword dictionary and KC-IDC files. We can obtain a list of documents containing this keyword through the KC-IDC index. To facilitate the description of its structure, we use Fig. 1 for illustration. The figure shows a KC-IDC index entry corresponding

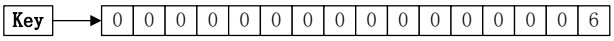

Fig. 1 KC-IDC index structure.

to the keyword "key". A 16-bit string is used to represent the ID of the document containing this keyword, where "6" denotes that the keyword "key" appeared six times in this document.

3.4 Symbol definition

The descriptions of the main symbols used in this article are shown in Table 1.

4 System Mode

4.1 Model introduction

The system model is shown in Fig. 2 and is mainly composed of four entities: DO, DU, public blockchain, and private blockchain.

(1) DO: The DO is mainly responsible for encrypting documents and indexes. Documents are encrypted and then uploaded to the public blockchain. Indexes are encrypted and uploaded to the private blockchain, together with the access control request.

(2) Private blockchain: This is used to store encrypted indexes and access control requests. After receiving encrypted indexes and access control requests,

Table 1 Symbol definitions.

Symbol	Description
DB	Document database
$PRF\{G, F\}$	Secure pseudorandom function
(u, s)	Keyword status
ptr_i, ptr_{i+1}	Index pointer and the next index pointer
t_w^1, t_w^2	Search token and key token
id_i, C_{id_i}	File index and encrypted index
L_R, α_w	Search list and random number
$Hash$	Hash function
H	Hash function used to generate token
$Token$	Access control token
RID	Request information ID
SID	Data sender ID
$REID$	Data recipient ID
w	Keyword
Map	State mapping
EDB	Encrypted index database
ST, N, Y	Search status. ST: Did nothing; N: Not searched; Y: Searched
U_i, U_j	Data owner and data user
P_α, V_α	Random number's encrypted index and the corresponding pointer
c	Ciphertext of information RI
PK	Hash function key

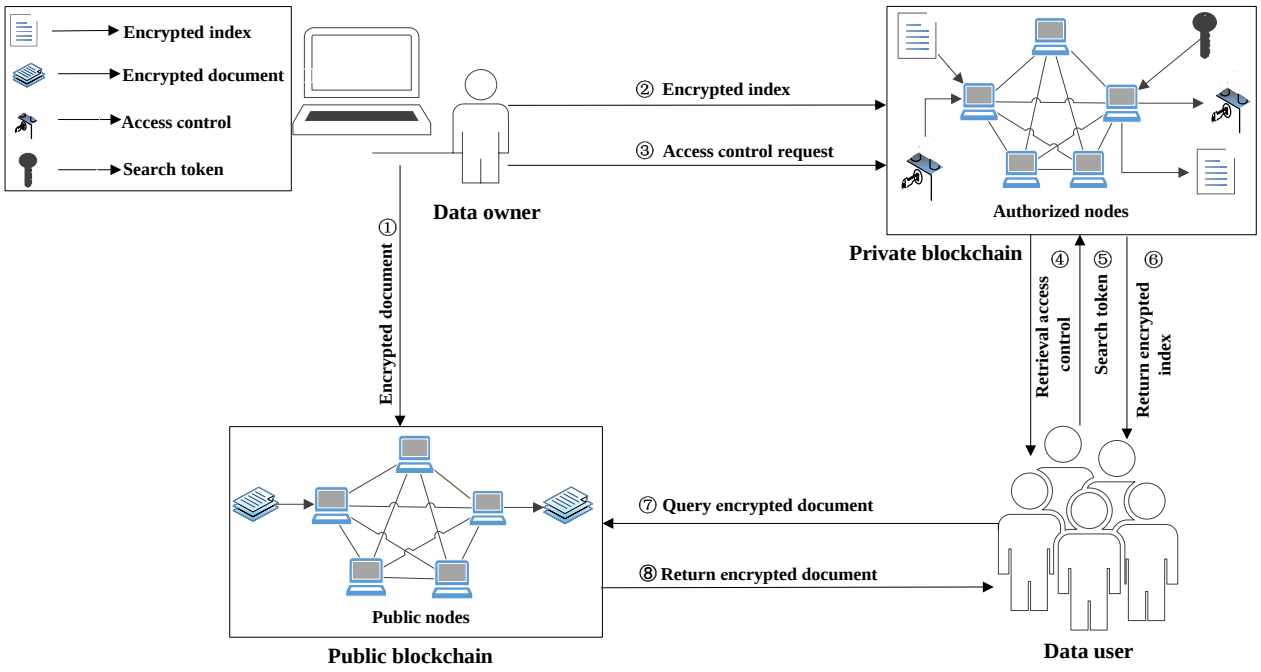

Fig. 2 System model.

the private blockchain retrieves the access control information for the DU. After checking and verifying the corresponding DU, the private blockchain executes a corresponding query in the private blockchain according to the search token sent by the received DU and, finally, returns the queried encrypted index to the DU.

(3) Public blockchain: This blockchain is used to store encrypted documents. After the public blockchain receives the request to retrieve the encrypted document sent by the DU, it performs the search in the public blockchain and returns the retrieved encrypted document corresponding to the encrypted index to the DU.

(4) DU: After receiving the access control information sent by the private blockchain, the DU verifies it and compares it with its own token[28] and request information. If both are equal, then it is the corresponding DU. Then, the DU sends its own search token to the private blockchain, queries the corresponding encrypted index, obtains the index results, and sends the request to retrieve the document to the public blockchain to receive the encrypted document sent by the public blockchain. Otherwise, the search is continued.

All search, add, and delete operations in the PPSE are performed through smart contracts, which have the characteristics of decentralization, low cost, high timeliness, and high accuracy.

4.2 Threat model

We believe that the DO and DU authorized by access control and the index and smart contract in Ethereum are all credible. To deal with more complex situations, we assume that there are potential threats from three aspects: blockchain, DU, and access control mechanism:

(1) The public blockchain has openness and transparency, and the stored encrypted data and operations are publicly visible. There may be potential attackers analyzing data and operations to find the connection between them, which threatens data security and user privacy.

(2) Assuming that DUs who are not authorized by access control are semi-trusted, then they may infer some sensitive information from the query results to obtain other data information.

(3) The access control mechanism must also ensure its nonconnectability. For any two outgoing transactions, it is impossible to prove that they are sent to the same person.

4.3 Security goal

Based on the above threat model, we should adhere to the following security design to protect user privacy and data security.

(1) Forward privacy: Forward privacy means that the DO uploads a new encrypted document. This document contains keywords that have been previously searched. The attacker cannot search for the document through the previous search trapdoor. It is impossible to obtain the relationship between keywords and documents. A new keyword tag must be used every

time a new file is added so that the association between different queries can be protected.

(2) Backward privacy: Backward privacy means that after a document that previously existed is deleted, one can no longer search for any information about it. In addition to the query result and update time, the solution cannot obtain any information, so the backward privacy is Type II.

(3) Integrity: With the application of blockchain technology, blockchain data are immutable and irrevocable to prevent data leakage and attackers from modifying them and achieve the goal of protecting data integrity.

(4) Privacy: The data on the blockchain are open and transparent, and other potential attackers will analyze these data and indexes. Therefore, it is necessary to encrypt data and indexes in advance and introduce an access control mechanism through smart contracts to restrict the access of DUs and prevent attackers from obtaining any sensitive information.

5 System Description

5.1 Design concept

The forward privacy attribute ensures that the attacker cannot search the newly added encrypted document through the previous search trapdoor. To achieve this goal, we generate new keyword labels during updating so that different queries are not related to each other. However, this method conflicts with the integrity of the result verification. When purely relying on the DO to verify the query results, storage and calculation costs are quite high. In addition, if the encrypted index and encrypted data are outsourced to the server, then the query results are uncontrollable.

Based on the above problems, first, we store data in a public distributed system so that the integrity and correctness of queries can be guaranteed by the trust maintained by a distributed entity. The underlying technology blockchain can be implemented like a decentralized system. Second, when using blockchain technology, there is no need to trust the operations performed by a third party (i.e., cloud server), and DOs or users do not suffer losses due to potential data inaccessibility. Finally, we store the encrypted index in the private blockchain while outsourcing the corresponding encrypted documents to the public blockchain. The consensus function of the blockchain inherently performs the verification operation, and the DU can efficiently retrieve the index while ensuring data correctness. At this point, the conflict between forward privacy and result verification and the problem of uncontrollable query results can be resolved.

Backward privacy ensures that after a previously added document is deleted, one can no longer search for any information about it. The PPSE uses the KC-IDC index structure and AES-128 encryption algorithm. When adding and deleting operations, the search token and key token are updated dynamically. The adversary cannot recognize the difference between the delete operation and the add operation, so it cannot know the relationship between the keywords and deleted documents to achieve backward privacy.

In addition, all the search, add and delete operations of the PPSE solution are implemented using Ethereum smart contracts[13]. All operations can be executed automatically, efficiently, and credibly, which makes data sharing convenient. Each transaction and data point in Ethereum is publicly visible, which may lead to data leakage. To protect data privacy, an access control mechanism is designed and embedded in the private blockchain. DUs are authenticated twice to restrict their access to the private blockchain. Only users authorized by access control can access the private blockchain. After verification, the private blockchain executes the query operation after the authorized user and returns the corresponding encrypted index to the DU. After the DU obtains the encrypted index, it sends a request to query the encrypted document to the public blockchain, and the public blockchain returns the encrypted document to the DU after the query operation is performed. The private blockchain has a fast transaction speed and high query efficiency. The index and data are stored separately, and encrypted documents are located through the encrypted index to reduce the storage overhead on the blockchain and improve solution performance.

Using access control tokens and applying them to the blockchain provide the following benefits:

(1) We adopt a token-based access control model. This model uses the token as the data body to record the rules and logic of access control and perform the authentication processing of access control, which can guarantee the correct subject, access, and object. It has certain comparative advantages in terms of security, credibility, circulation, and concurrency.

(2) Two verifications are used in this access control scheme. First, we need to verify whether the token is

consistent to guarantee that it is received accurately. Second, the DU's ID is verified to guarantee that it is an authorized recipient. The two types of verification make the scheme more accurate.

(3) The access control scheme is applied to the blockchain to limit the access rights of DUs to the private blockchain and prevent data information leakage. This method can help avoid hidden dangers caused by security vulnerabilities in the blockchain network, effectively resist individuals and collusive malicious users, and protect data privacy.

5.2 Specific plan

5.2.1 Four basic algorithms

Setup (1^λ): The DO initializes the system locally, inputs the parameter λ, and outputs the master key $k \leftarrow \{0,1\}^\lambda$. The empty sets Map and EDB are initialized, where Map is used to record the updating and search status. The PPSE scheme uses the KC-IDC index structure, which is stored in the encrypted index set EDB in the form of (w, id) pairs, where w and id are the keyword and file identifier, respectively.

As shown in Algorithm 1, the keyword w and file identifier id are inputted, the update times and search times are set to 0, and the search status is set from ST to N to indicate that it has not been searched. For each keyword, first, the search token t_w^1 and key token t_w^2 are generated, and then, the search token is used to generate a guide, ptr_i, for each index. When the first pointer is initialized, there is no previous pointer, the previous pointer is set to null, and then, the XOR(exclusive OR) operation is performed to obtain the current encrypted index (P_i) and its corresponding pointer (V_i), which are stored as a piece of data in the encrypted index database EDB and updated. The state corresponding to the keyword is stored in Map. Finally, the encrypted index EDB is uploaded to the private blockchain by adopting the smart contract. The DO updates Map locally.

Search (k, Map, G, w)**:** As shown in Algorithm 2, the private blockchain forwards the received access control information to the DU, who then sends a search token to the private blockchain. After calculation, user U_j is determined as an authorized user. Then, the smart contract is called on the private blockchain to perform the search operation. First, an empty list L_R is created to obtain the status information of the keyword. The DU generates a new token for the search keyword and judges ST. If it is equal to Y, then the keyword has not been updated after the search; otherwise, the updated index of this keyword has not yet been searched, and the latest index pointer, ptr_i, needs to be calculated.

Subsequently, U_j sends the search token to the private blockchain after receiving the access control information. After calculation, U_j is determined as an authorized user. The private blockchain calls the smart contract to search for the encrypted index related to keyword w until the index pointer corresponds to a null value, and an empty result list L_R is created. Then, all the ciphertexts and random number results found in the query are $L_R = \{C_{id_i}, \ldots, C_{id_n}, \alpha_w\}$ and are returned

Algorithm 1 Buildindex ($w, k, Map, EDB, PRF\ \{G, F\}, ptr, id$)

Input: master key k, security $PRF\ \{G, F\}$, index pointer ptr, state mapping $Map \leftarrow \{u, s\}$, keyword w, and file identifier $id_i, i \in \{1, \ldots, n\}$

Output: EDB and Map

Step R1: For each w, U_i generates a random number α_w and sets up $\{u, s\} \leftarrow 0$; $S \leftarrow N$; U_i computes tokens
$t_w^1 \leftarrow G(k, w||u||1)$;
$t_w^2 \leftarrow G(k, w||u||1)$;
$ptr_{(n+1)} \leftarrow F(w, \alpha_w)$.

Step R2: For $\{id_1, \ldots, id_n\} \in DB(w)$, U_i computes $ptr_i \leftarrow F(w||id_i)$ and $ptr_{i+1} \leftarrow F(w||id_{i+1})$.

Step R3: U_i computes $C_{id_i} \leftarrow Enc(k, id_i)$.

Step R4: U_i computes $P_i \leftarrow G(t_w^1, ptr_i) \oplus ptr_{i+1}$; $V_i \leftarrow G(t_w^2, ptr_i) \oplus C_{id_i}$.

Step R5: U_i stores $[ptr_i: P_i, V_i]$ in the encrypted index database EDB of the private blockchain, i++.

Step R6: For random number α_w, U_i computes $P_\alpha \leftarrow G(t_w^1, ptr_{n+1}) \oplus \perp$ and $V_\alpha \leftarrow G(t_w^2, ptr_{n+1}) \oplus \alpha_w$.

Step R7: U_i stores $[ptr_{n+1}: P_\alpha, V_\alpha]$ in the encrypted index database EDB of the private blockchain.

Step R8: U_i stores $[w: ptr_1, u, s]$ in the table Map and uploads the EDB to the private blockchain by adopting the smart contract.

Step R9: Private blockchain updates $EDB\ [ptr_i] \leftarrow \{P_i, V_i\}$.

Algorithm 2 Search ($k, Map, PRF\ \{G\}, w$)

Input: master key k, security $PRF\ \{G\}$, search keyword w, state mapping Map

Output: search result $L_R = \{C_{id_1}, \ldots, C_{id_n}, \alpha_w\}$

Step R1: U_j obtains $\{ptr_1, u, s\} \leftarrow Map[w]$, and sets up $S \leftarrow Y$.

Step R2: U_j computes tokens $t_w^1 \leftarrow G(k, w||u||1)$ and $t_w^2 \leftarrow G(k, w||u||1)$.

Step R3: U_j sends $\{t_w^1, t_w^2, ptr_1\}$ to the search smart contract and broadcasts it in the private blockchain.

Step R4: For $i = 1$ to $ptr_i == \perp$: the private blockchain gets $P_i, V_i \leftarrow EDB[ptr_i]$ and computes $ptr_{i+1} \leftarrow G(t_w^1, ptr_i) \oplus P_i$; $C_{id_i} \leftarrow G(t_w^2, ptr_i) \oplus V_i$, i++.

Step R5: Return $L_R = \{C_{id_1}, \ldots, C_{id_n}, \alpha_w\}$ to the corresponding DU.

Step R6: Data user U_j searches for relevant encrypted documents in the public blockchain according to the index.

to the corresponding DU U_j.

Addition($k, G, Hash, Map$): As shown in Algorithm 3, the DO obtains the update status corresponding to the keyword w. If S is equal to Y, then it represents the first update after the keyword search and that the number of searches that need to be updated is s. The new key is used to generate a new token and index pointer. Finally, S is set to N. Otherwise, it means that the key has not been searched after the update, and the previous key k and token can be used. The new index is identified after the pointer is connected to the previous pointer. After obtaining the encrypted data, the hash function operation is performed, and the obtained hash value is used as input to call the smart contract, adding the data to the private blockchain and uploading the corresponding encryption to the public blockchain at the same time. Then, the DO updates Map.

Delete (*blockNo*, *DB*(*w*), *EDB*): As shown in Algorithm 4, the DO uses the block number $blockNo$ as an input to call the deleted smart contract and execute the corresponding action according to the input block number. During the data deletion operation, the DO also deletes the encrypted documents in the public blockchain.

Algorithm 3 Addition (*k*, *G*, *Hash*, *Map*)

Input: master key k, security $PRF\{G\}$, hash function $Hash$, State mapping Map

Output: update index I and block number $blockNo$

Step R1: For w, $\{id_1, \ldots, id_m\} \in DB(w)$: U_i generates a random number r and g.

Step R2: U_i obtains $\{ptr_i, u, s\} \leftarrow Map[w]$ and computes $C_{id_i} \leftarrow Enc(k, id_i)$.

Step R3: If ST==Y, then s=s+1; $t_w^1 \leftarrow G(k, w||u||1)$; $t_w^2 \leftarrow G(t_w^2, w||u||1)$.

Step R4: U_i computes $P_i \leftarrow G(t_w^1, ptr_i) \oplus \perp$; $V_i \leftarrow G(t_w^2, ptr_i) \oplus C_{id_i}$; and sets up $ST \leftarrow N$.

Step R5: Else, U_i computes $t_w^1 \leftarrow G(k, w||u||1)$; $t_w^2 \leftarrow G(k, w||u||1)$; $P_i \leftarrow G(t_w^1, ptr_i) \oplus ptr_{i+1}$; $V_i \leftarrow G(t_w^2, ptr_i) \oplus C_{id_i}$.

Step R6: U_i computes $EDB[ptr_i] \leftarrow P_i, V_i; hash \leftarrow Hash(PK, g, EDB[ptr_i], r)$.

Step R7: U_i adopts smart contract to upload the added index to the private blockchain, and updates $Map[w] \leftarrow \{ptr_i, u, s\}$.

Step R8: The private blockchain updates $EDB[ptr_i]$.

Step R9: At the same time, the public blockchain updates encrypted documents.

Algorithm 4 Delete (*blockNo*, *DB*(*w*), *EDB*)

Input: block number $blockNo$

Step R1: For each w, U_i adopts the deleted smart contract: for $\{id_1, \ldots, id_m\} \in DB(w)$: delBlock($blockNo$).

Step R2: The private blockchain updates the encrypted index database $EDB[ptr_i]$.

Step R3: At the same time, the public blockchain updates the encrypted documents.

5.2.2 Access control scheme

The access control scheme involves the DO sending an access control request to the private blockchain using a smart contract. As shown in Fig. 3, the access control request contains the request information and the token. Request information RI: $\{RID, SID, REID\}$, where RID stands for the request ID, SID stands for the the DO's ID, and $REID$ stands for the data receiver's ID. After the private link receives this access control request, it forwards it to each DU who wants to access the private link, and the DUs perform two verifications after receiving the access control information. First, they verify whether they received the token and that it is equal to its own $Token'$, and if they are equal, then DUs verify whether the $REID$ in the request information is equal to its own $REID$. If both verifications are correct, then it means that the users are authorized DUs, as shown in Fig. 4.

(1) **Access control token generation**: As shown in Algorithm 5, first, U_i needs to obtain the private key K_{u_j} of U_j and its own random number r, calculate the $Token$, and use the private key K_{u_j} of U_j to make a

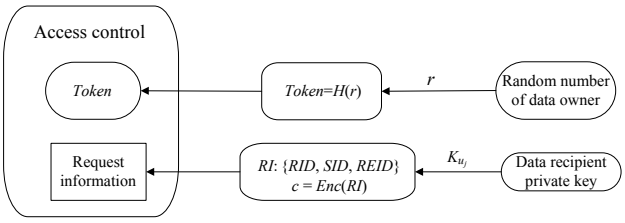

Fig. 3 Access control token generation.

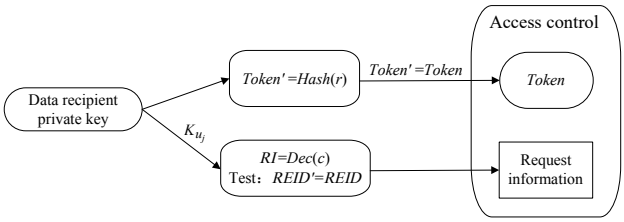

Fig. 4 Retrieve access control token.

Algorithm 5 Access control token generation

Input: U_i obtains the private key K_{u_j} of U_j, hash function $\{H\}$, request information RI {*RID*, *SID*, *REID*}, and random number r

Output: $Token$ and ciphertext c

Step R1: U_j computes $Token = H(r)$ and $c = Enc(RI)$.

Step R2: Return $Token$ and c.

Step R3: U_i broadcasts c and $Token$ as transactions on the Ethereum private blockchain.

request. The information RI is encrypted to obtain the ciphertext c, and the DO broadcasts c and the $Token$ as a transaction in the Ethereum private blockchain.

(2) **Retrieval of the access control token**: As shown in Algorithm 6, U_j checks the latest transaction in the newly generated block and extracts the access control request information c and $Token$. The DU first calculates the $Token^{'}$ and determines whether the received $Token$ is equal to it. If they are equal, then the self-decryption key K_{u_j} is restored, ciphertext c is decrypted, and the obtained $REID$ is compared with its own ID. At this time, the comparison ID is also equal, and then, the user is considered an authorized user. Otherwise, the next user is compared.

6 Security Analysis

6.1 Security model

The security model of the solution is completed by the real model Real and the ideal model Ideal. This solution has the same behavior as the real model. The ideal model reflects the behavior of the simulator S.S uses the leakage function $L = (L^{Setup}, L^{Update}, L^{Search})$ as the input. There are differences between the Buildindex algorithm, add algorithm, and delete algorithm used in the solution. However, the behavior is the same for adversary A. Therefore, the leakage functions of the three are uniformly represented by L^{Update}, L^{Setup} is the leakage during initialization, and L^{Search} is the leakage during the search. Real and Ideal games are defined as follows:

$Real_A(\lambda)$: First, the Setup algorithm is run to output the encrypted database EDB. Adversary A executes the search query sh or updates the query (op, up), where op represents the update operation, up represents the updated (w, id) pair, and then the adversary A outputs the result $b \in \{0, 1\}$.

$Ideal_{(A,S)}(\lambda)$: The simulator S first executes the leakage function L^{Setup}. The adversary A executes the search query sh or update query (op, up). The simulator S uses the leakage functions L^{Update} and L^{Search} as the input and returns the result to A, and the adversary A outputs the result $b \in \{0, 1\}$.

Algorithm 6 Retrieval of the access control token

Input: $Token$ and c

Step R1: U_j computes $Token^{'} = Hash(r)$ and compares $Token^{'} \stackrel{?}{=} Token$.

Step R2: U_j restores the corresponding decryption key K_{u_j}.

Step R3: U_j decrypts c to request information RI: $\{RID, SID, REID\}$.

Step R4: U_j computes its own ID and the $REID$ in the request information. Then U_j compares $REID^{'} \stackrel{?}{=} REID$.

Step R5: The two comparisons are equal, and the output "This user U_j is an authorized user" is displayed.

Definition 1 If there is an efficient simulator S and input L for any probabilistic polynomial adversary A, such that

$$|Pr[Real_A(\lambda) = 1] - Pr[Ideal_{(A,S)}(\lambda) = 1]| \leqslant negl(\lambda) \tag{1}$$

then the L-adaptive-secure SSE scheme is forward private. Among them, negl(λ) is a function that can be ignored, indicating that this scheme is L-adaptive-secure.

6.2 Security definition

6.2.1 Forward privacy

If no adversary can obtain any information of the previous search query through the update, then this situation satisfies forward privacy[11].

Definition 2 If the update leakage function L^{Update} can be written as

$$L^{Update}(op, up) = L'(op, (id_i, u_i)) \tag{2}$$

then the L-adaptive-secure symmetric searchable encryption scheme is forward privacy.

The leakage function of this scheme can be written as

$$L^{Update}(op, w, id) = L'(op, id) \tag{3}$$

6.2.2 Backward privacy

Backward privacy ensures that after a previously added document is deleted, any information about it cannot be found when searching. In this solution, if the previously added data (w, id) is deleted, then adversary A can no longer query the file identifier id. Then, the PPSE solution is backward privacy. Backward privacy is divided into three different levels, and the PPSE scheme achieves Type II backward privacy[11].

Type II: In addition to the the number and type of previous updates associated with w, the identifiers of files containing w currently in the database, Type II schemes also reveal when all updates related to w took place.

Definition 3 If the search and update leakage functions L^{Update} and L^{Search} can be written as

$$L^{Update}(op, w, id) = L'(op, w) \tag{4}$$

$$L^{Search}(w) = L''(TimeDB(w), Updates(w)) \tag{5}$$

where L' and L'' represent two stateless functions, then the L-adaptive-secure SSE scheme is backward private.

$TimeDB(w)$ is the list of all documents matching w, excluding the deleted ones, together with the timestamp of when they were inserted in the database. Formally, $TimeDB(w)$ can be constructed from the query list Q as follows:

$$TimeDB(w) = \{(qt, id) | (qt, add, (w, id)) \in Q, \forall qt', (qt', del, (w, id)) \notin Q\} \quad (6)$$

where qt is the query timestamp. $DB(w) = \{id | \exists qt,$ s.t., $(qt, id) \in TimeDB(w)\}$. Note that $TimeDB()$ is completely oblivious to any document added to $DB(w)$ that is later removed, but retains all other information. As such, $TimeDB()$ captures a strong notion of backward privacy revealing only the time of the insertion of the documents currently containing the search query w. $Updates(w)$ is a function that returns the timestamp of all insertion and deletion operations for w in Q. Formally, $Updates(w) = \{qt | (qt, add, (w, id)) \in Q$ or $(qt, del, (w, id)) \in Q\}$.

Let us demonstrate the differences between these notions with an example. Consider the following sequence of updates, in the order of arrival: $(add, id_1, w_2), (add, id_2, w_1), (del, id_1, w_1), (add, id_3, w_2)$. The document corresponding to the keyword w_1 is deleted after being added. Let us consider the leakage for each definition after a search query on w_1. Type II reveals id_1, and this entry was added at $time_1$. It also reveals that there were a total of three updates for w_1 and shows the time of each update.

6.3 Specific security analysis

In this section, a theorem is given to prove the security of the scheme.

For forward privacy, every time the PPSE scheme updates the data, the hash function will produce different outputs for different inputs and generate new keyword tags. First, whether the keyword has been searched and whether the previous search token can be used when it has not been searched are determined. If a search operation has been performed, then a new search token and key token need to be generated to ensure that the newly added document will not be associated with the previous search token, thereby achieving forward privacy. For backward privacy, the PPSE scheme uses the KC-IDC index and uploads the encrypted data to the blockchain. According to the semantic security of the hash function, the probability distribution of the hash value is indistinguishable. In particular, when r is randomly selected, the $Hash(PK, g, C_{id_i}, r)$ function cannot get any information about the data C_{id_i}. The solution uses the encrypted index form of $C_{id_i} \leftarrow G(t_w^2, ptr_i) \oplus V_i$. The ptr_i pointer is stored locally by the DO. Even if the blockchain is open, adversary cannot be obtained the location information of the text. However, storing data in chronological order will leak the update time, so the degree of leakage is smaller, that is, it reaches Type II backward privacy. The proof is as follows:

Theorem 1 If the pseudo-random functions G and F are safe and all hash functions have anti-collision properties, then this solution satisfies L-adaptive-secure, forward private, and Type II backward private schemes. We define the leakage function of this scheme as $L = (L^{Update}, L^{Search}), L^{Update}(op, w, id) = L'(op, w)$, $L^{Search}(w) = L''(TimeDB(w), Updates(w))$.

Proof This solution uses the terms Real and Ideal to set up a series of games and uses the leak function of Theorem 1 to simulate Ideal. Accordingly, adversary A cannot distinguish between Real and Ideal, and the security of the solution is proven.

G_0 is the same as the real model game $Real_A(\lambda)$, with

$$|Pr[G_0 = 1] = Pr[Real_A(\lambda) = 1]| \quad (7)$$

G_1: When the key of keyword w is generated during the updating process, if the key has been searched before, then another key is randomly generated through the G function. Because the pseudorandom function G is secure, G_1 and G_0 are indistinguishable.

$$|Pr[G_1 = 1] = Pr[G_0 = 1]| \quad (8)$$

G_2: The hash function H^* is transformed into a random oracle, and the encrypted index pointer is generated by the random oracle. In addition, G_2 is the same as G_1, and the encrypted index pointer is randomly selected. In this case, the probability that adversary A can guess the correct encrypted index pointer is $1/2^\lambda$. We assume that adversary A can perform polynomial queries q times. Then, the probability is $q/2^\lambda$ because the hash function generates the value probability. The distribution is indistinguishable, and it is more resistant to collision. Thus, for adversary A, the advantages of G_2 and G_1 can be distinguished as follows:

$$|Pr[G_2 = 1] - Pr[G_1 = 1]| \leqslant q/2^\lambda \quad (9)$$

G_3: The last game is $Ideal_{(A,S)}(\lambda)$. The simulator uses the leak function to simulate from the perspective of adversary A, where the leak function includes L^{Update} and L^{Search}. From the perspective of adversary A,

G_3 and G_2 behave the same, so they are considered indistinguishable.

$$|Pr[G_3 = 1] = Pr[G_2 = 1] = Pr[Ideal_{(A,S)}(\lambda) = 1]| \quad (10)$$

Based on the above analysis, the situation can be summarized as follows:

$$|Pr[Real_A(\lambda)=1]-Pr[Ideal_{(A,S)}(\lambda)=1]| \leqslant q/2^{\lambda} \quad (11)$$

Because the pseudorandom function used in the PPSE scheme is secure, the hash function is semantically secure, the probability distribution of the hash value is indistinguishable and has anti-collision properties, and trapdoor collisions make the scheme highly secure. Hence, the probability of adversary A distinguishing the real and ideal models can be regarded as λ. In this case, the two models are indistinguishable.

The certificate is complete. ■

7 Formula Analysis

7.1 Performance analysis

We compare the performance of the PPSE scheme with the schemes in Refs. [8, 9, 21–23]. The results are shown in Table 2, which mainly analyzes the search and updating complexity, local storage overhead, and security of the solution. Among them, in some studies[8, 9], the data are stored on the cloud server, and their search and update complexity is lower than those of the PPSE scheme and those in Refs. [21–23], which store data on the blockchain. The performance gap is within an acceptable range, and storing the data on the blockchain can prevent the data from being changed and ensure the integrity of the query results. Therefore, it improves the security and privacy of the data. Because the PPSE scheme uses the KC-IDC index and access control information for easy calculation, its efficiency is higher than that of the schemes in Refs. [21–23] without reducing security.

Among them, a_w is the total number of updates, K is the number of different keywords, D is the number of documents, T is the total pairs of (w, id), B is the number of blocks, $d_{\max}$ is the maximum number of documents, $w_{\max}$ is the maximum number of keywords, and n_w is the number of searches performed for keyword w.

This solution uses the KC-IDC index structure. The complexity of extracting and constructing an index from the original dataset is $O(\log T)$, but it has nothing to do with the document size during retrieval, so the complexity of querying keywords is $O(1)$. Because the data are stored on the blockchain, the first block is checked forward from the latest block B, one by one, during the query, and the query time complexity is $O(B)$ for each visit. For the query operation with a search result of n_w, the query time complexity is $O(n_w B)$ when this solution executes n_w visits in parallel. When performing an updating operation, the add and delete operations execute a fixed number of blocks, so the time complexity of the update is $O(B)$. For the storage overhead, the DO wants to store as little data as possible. The content that needs to be stored is the encryption key k and state Map, so the local storage is $O(1)$, and the encrypted database EDB needs to be stored on the blockchain. In addition, the overhead of the access control information is $O(B)$.

7.2 Experiment analysis

To demonstrate the security and unique performance characteristics of this design, a simulated Ethereum network is built locally. The simulated network is very similar to the real Ethereum environment, except that its mining block time is set to 0. This method allows us to focus on the performance of the search part of the smart contract, regardless of the time-consuming mining process and complex network environments in Ethereum, such as broadcast delays and transaction mining delays. The environment is based on a Windows 10 (64-bit) system, and the specific hardware configuration is an Intel (R) Core (TM) i5-10400F CPU @2.90 GHz

Table 2 Performance comparison.

Scheme	Blockchain	Search	Update	Forward privacy	Backward privacy	Local storage
Fides[8]	No	$O(a_w)$	$O(1)$	Yes	II	$O(K \log D)$
Horus[9]	No	$O(n_w \log d_w \log T)$	$O(\log^2 T)$	Yes	III	$O(K \log D)$
Orion[9]	No	$O(n_w \log^2 T)$	$O(\log^2 T)$	Yes	I	$O(1)$
$\prod_{fair}$[21]	Yes	$O(n_w \times B \times d_w)$	$O(B)$	Yes	No	$O(1)$
Chen et al.[22]	Yes	$O(n_w \times B \times d_w)$	$O(B)$	Yes	No	$O(1)$
Jiang et al.[23]	Yes	$O(n_w \times B \times d_{\max} w_{\max})$	$O(B)$	Yes	No	$O(1)$
PPSE	Yes	$O(n_w \times B \times d_{\max})$	$O(B)$	Yes	II	$O(1)$

processor with 16 GB of memory. The Enron email dataset is used as the original dataset, and a subset is extracted from it as the test dataset. In the experiment, the smart contract uses the solidity language, and the language for interacting with the smart contract is Python.

The simulation experiment of the PPSE scheme mainly tests for the efficiency of the search and updating stages and evaluates the search time by returning a fixed number of matching documents. The updating cost is given by adding and deleting a fixed number of files. The PPSE scheme is compared with the Fides[8] of the same backward private level and Orion[9] with higher backward private level than PPSE in the cloud environment and an existing study[23] in the blockchain environment that also stores indexes and documents separately. The results are analyzed as follows.

To show the core algorithm, the number of matching documents is set to increase from 100 to 500 for searching. Figure 5 shows the increasing trend of the search time as the number of matching documents changes. We execute each plan 50 times and then take the average. The more matching documents there are, the slower the search algorithm. The larger the dataset in the cloud environment, the more data to be retrieved, and the greater the search time. Similarly, there are also blocks to be mined on the blockchain. The greater the number is, the longer the loading time, which makes the algorithm execution efficiency lower than that of the cloud environment. However, Fig. 5 shows that the query efficiency of the PPSE scheme and those of the Fides[8] and Orion[9] schemes in the cloud environment are not much different, but the security is significantly higher in the former, as compared to a study on the same blockchain environment[23]. The query efficiency advantage is evident. In addition, to read data from Ethereum, there is no need to exert great efforts because we hold a copy of the Ethereum database. On the blockchain, search operations are performed through smart contracts to ensure data security.

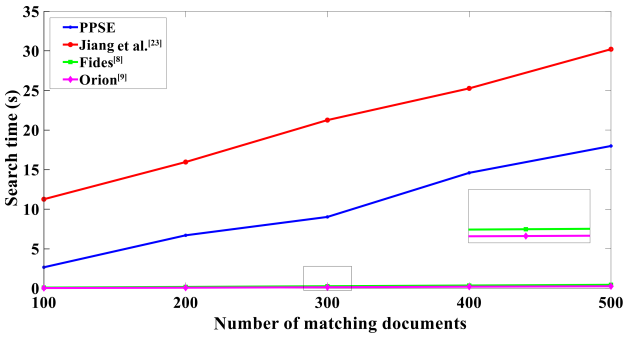

Fig. 5 Search time comparison.

In Figs. 6–8, the consumption and time costs that change with the number of files added or deleted are described. The execution time and gas consumption linearly increase with the number of files added and deleted (Gas is a unit used to measure the amount of computational work required to perform a specific operation on the Ethereum blockchain). By selecting files of different sizes, the keyword index pair in the file is increased from 20 to 100 for testing. Figure 7 shows that the updating efficiency of the Fides[8] and Orion[9] schemes in the cloud environment is not much different than that of the PPSE scheme, but the security is significantly lower. Compared with Ref. [23] in the same blockchain environment, the PPSE scheme has

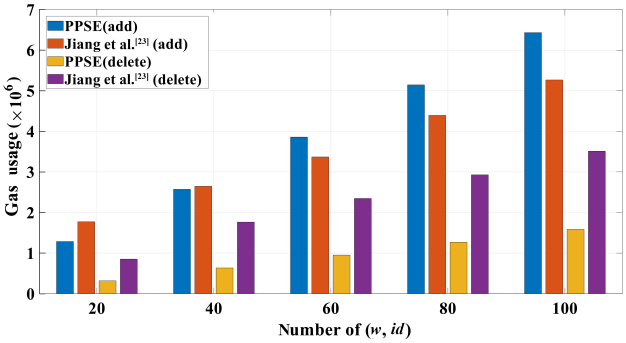

Fig. 6 Gas consumption comparison.

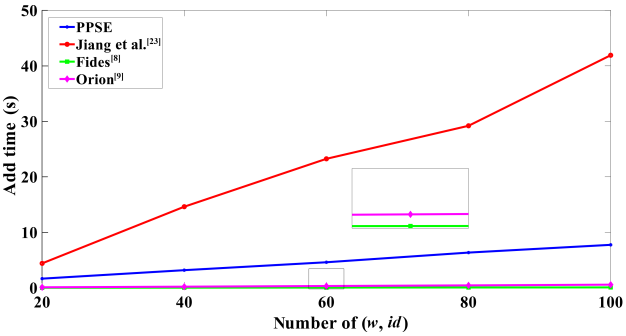

Fig. 7 Add time comparison.

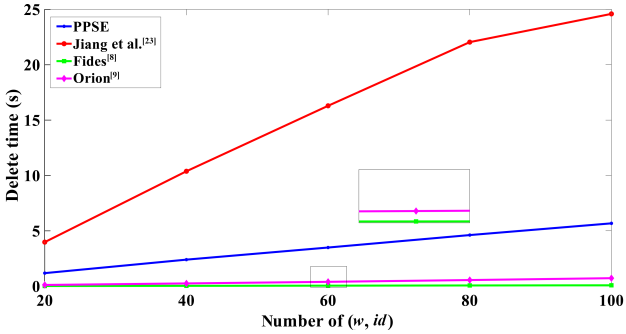

Fig. 8 Delete time comparison.

evident update efficiency advantages. The additions and deletions performed on the blockchain are all performed through smart contracts, which are decentralized and have high timeliness, whereas the cost is low, the accuracy is high, and the security is much higher than that of the cloud environment.

8 Conclusion

Traditional privacy-preserving SE schemes rely on cloud servers to complete search operations. In this study, based on blockchain technology, a decentralized security model is designed to solve the problems of original malicious servers and attacks from malicious users. Compared with the existing verification schemes, the PPSE does not require the DO to verify, nor does it require them to send the results to a third party for verification. Using the blockchain itself to store data can obtain correct and immutable results. At the same time, we store the encrypted index in the private blockchain while outsourcing the corresponding encrypted documents to the public blockchain, and the access control mechanism is introduced to improve the query efficiency and the security of the encrypted data. The safety analysis indicates that the scheme meets the safety requirements, and the experimental results obtained concerning our prototype demonstrate the practicability of our scheme.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61972073), the Key Program of Natural Science Foundation of Hebei Province of China (No. F2019201290), and the Natural Science Foundation of Hebei Province of China (No. F2018201153).

References

[1] D. X. Song, D. Wagner, and A. Perrig, Practical techniques for searches on encrypted data, in *Proc. 2000 IEEE Symp. on Security and Privacy. S&P 2000*, Berkeley, CA, USA, 2000, pp. 44–55.

[2] Z. T. Guan, N. Y. Wang, X. F. Fan, X. Y. Liu, L. F. Wu, and S. H. Wan, Achieving secure search over encrypted data for e-commerce: A blockchain approach, *ACM Trans. Int. Technol.*, vol. 21, no. 1, p. 12, 2021.

[3] H. Y. Li, T. Wang, Z. R. Qiao, B. Yang, Y. Y. Gong, J. Y. Wang, and G. Y. Qiu, Blockchain-based searchable encryption with efficient result verification and fair payment, *J. Inf. Secur. Appl.*, vol. 58, p. 102791, 2021.

[4] Z. L. Liu, T. Li, P. Li, C. F. Jin, and J. Li, Verifiable searchable encryption with aggregate keys for data sharing system, *Future Gener. Comput. Syst.*, vol. 78, pp. 778–788, 2018.

[5] S. Kamara, C. Papamanthou, and T. Roeder, Dynamic searchable symmetric encryption, in *Proc. 2012 ACM Conf. on Computer and Communications Security*, Raleigh, NC, USA, 2012, pp. 965–976.

[6] E. Stefanov, C. Papamanthou, and E. Shi, Practical dynamicsearchable encryption with small leakage, in *Proc. of Network and Distributed System Security Symposium*, San Diego, CA, USA, pp. 72–75, 2014.

[7] R. Bost, $\sum o\varphi o\varsigma$: Forward secure searchable encryption, in *Proc. 2016 ACM SIGSAC Conf. on Computer and Communications Security*, Vienna, Austria, 2016, pp. 1143–1154.

[8] R. Bost, B. Minaud, and O. Ohrimenko, Forward and backward private searchable encryption from constrained cryptographic primitives, in *Proc. 2017 ACM SIGSAC Conf. on Computer and Communications Security*, Dallas, TX, USA, 2017, pp. 1465–1482.

[9] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, New constructions for forward and backward private symmetric searchable encryption, in *Proc. 2018 ACM SIGSAC Conf. on Computer and Communications Security*, Toronto, Canada, 2018, pp. 1038–1055.

[10] S. F. Sun, X. L. Yuan, J. K. Liu, R. Steinfeld, A. Sakzad, V. Vo, and S. Nepal, Practical backward-secure searchable encryption from symmetric puncturable encryption, in *Proc. 2018 ACM SIGSAC Conf. on Computer and Communications Security*, Toronto, Canada, 2018, pp. 763–780.

[11] S. Patranabis and D. Mukhopadhyay, Forward and backward private conjunctive searchable symmetric encryption, in *Proc. of 28th Annu. Network and Distributed System Security Symp.*, doi:10.14722/ndss.2014.23298.

[12] J. Li, Y. Y. Huang, Y. Wei, S. Y. Lv, Z. L. Liu, C. Y. Dong, and W. J. Lou, Searchable symmetric encryption with forward search privacy, *IEEE Trans. Depend. Secure Comput.*, vol. 18, no. 1, pp. 460–474, 2021.

[13] X. Q. Liu, G. M. Yang, W. Susilo, J. Tonien, X. M. Liu, and J. Shen, Privacy-preserving multi-keyword searchable encryption for distributed systems, *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 561–574, 2021.

[14] G. Asharov, G. Segev, and I. Shahaf, Tight tradeoffs in searchable symmetric encryption, *J. Cryptol.*, vol. 34, no. 2, p. 9, 2021.

[15] K. He, J. Chen, Q. X. Zhou, R. Y. Du, and Y. Xiang, Secure dynamic searchable symmetric encryption with constant client storage cost, *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1538–1549, 2021.

[16] Q. Y. Song, Z. T. Liu, J. H. Cao, K. Sun, Q. Li, and C. Wang, SAP-SSE: Protecting search patterns and access patterns in searchable symmetric encryption, *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1795–1809, 2021.

[17] A. Soleimanian and S. Khazaei, Publicly verifiable searchable symmetric encryption based on efficient cryptographic components, *Des. Codes Cryptogr.*, vol. 87, no. 1, pp. 123–147, 2019.

[18] Q. Y. Tong, Y. B. Miao, X. M. Liu, K. K. R. Choo, R. Deng, and H. W. Li, VPSL: Verifiable privacy-preserving data search for cloud-assisted internet of things, *IEEE Trans. Cloud Comput.*, doi: 10.1109/TCC.2020.3031209.

[19] W. Y. Yang and Y. S. Zhu, A verifiable semantic searching scheme by optimal matching over encrypted data in public cloud, *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 100–115, 2021.

[20] Q. Tang, Towards blockchain-enabled searchable encryption, in *Information and Communications Security*, J. Zhou, X. Luo, Q. Shen, and Z. Xu, eds. Cham, Switzerland: Springer, 2020, pp. 482–500.

[21] S. S. Hu, C. J. Cai, Q. Wang, C. Wang, X. Y. Luo, and K. Ren, Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization, in *Proc. of IEEE INFOCOM 2018–IEEE Conf. on Computer Communications*, Honolulu, HI, USA, 2018, pp. 792–800.

[22] L. X. Chen, W. K. Lee, C. C. Chang, K. K. Choo, and N. Zhang, Blockchain based searchable encryption for electronic health record sharing, *Future Gener. Comput. Syst.*, vol. 95, pp. 420–429, 2019.

[23] S. R. Jiang, J. Q. Liu, L. M. Wang, and S. M. Yoo, Verifiable search meets blockchain: A privacy-preserving framework for outsourced encrypted data, in *Proc. of 2019 IEEE Int. Conf. on Communications* (*ICC*), Shanghai, China, 2019, pp. 1–6.

[24] Y. Guo, C. Zhang, and X. H. Jia, Verifiable and forward-secure encrypted search using blockchain techniques, in *Proc. of 2020 IEEE Int. Conf. on Communications* (*ICC*), Dublin, Ireland, 2020, pp. 1–7.

[25] P. L. Li, H. X. Xu, T. J. Ma, and Y. H. Mu, Research on fault-correcting blockchain technology, (in Chinese), *J. Cryptol. Res.*, vol. 5, no. 5, pp. 501–509, 2018.

[26] X. Han, Y. Yuan, and F. Y. Wang, Security problems on blockchain: the state of the art and future trends, (in Chinese), *Acta Autom. Sin.*, vol. 45, no. 1, pp. 206–225, 2019.

[27] J. L. Sun, S. Huang, C. Y. Zheng, T. Y. Wang, C. Zong, and Z. W. Hui, Mutation testing for integer overflow in ethereum smart contracts, *Tsinghua Science and Technology*, vol. 27, no. 1, pp. 27–40, 2022.

[28] R. Z. Du, A. L. Tan, and J. F. Feng, An attribute-based encryption scheme based on unrecognizable trapdoors, *Tsinghua Science and Technology*, vol. 25, no. 5, pp. 579–588, 2020.

Ruizhong Du received the PhD degree in information security from Wuhan University, China in 2012. Since 1997, he has been working at the School of Cyberspace Security and Computer, Hebei University, China. He is currently the associate dean, a doctoral supervisor, and a professor at the School of Cyberspace Security and Computer, Hebei University. He is the secretary-general of the Hebei Cyberspace Security Society and the executive director of the Hebei Computer Society. His research interests include network security, edge computing, and trusted computing. He is a member of IEEE.

Caixia Ma received the BEng degree in information security from Hebei Normal University, China in 2020. She is currently a master student in cyberspace security at Hebei University, China. Her main research interest is searchable encryption technology.

Mingyue Li received the MEng degree in information security from Hebei University, China in 2019. She is currently a PhD candidate at the College of Cyber Science, Nankai University, China. Her research interests include network security and searchable encryption technology.