Journal of Basrah Researches (Sciences) **50**(1), 304 (2024)
DOI: https://doi.org/10.56714/bjrs.50.1.24

Performing Encrypted Cloud Data Keyword Searches Using Blockchain Technology on Smart Devices

Salim Sabah Bulbul [1], **Zaid Ameen Abduljabbar** [2,*]

[1]Directorate General of Education Basrah, Ministry of Education, Basrah, Iraq
[2]Department of Computer Science, College of Education for Pure Sciences, University of Basrah, Basrah, 61004, Iraq

A R T I C L E I N F O

Received 16 May 2024
Accepted 25 June 2024
Published 30 June 2024

K e y w o r d s :

I/O efficiency, blockchain network, symmetric primitives, searchable encryption (SE), backward-security, forward-security.

Citation: Salim S. B. , Zaid A. A., J. Basrah Res. (Sci.) 50(1), 304 (2024). DOI:https://doi.org/10.56714/bjrs.50.1.24

A B S T R A C T

Data owners seeking to boost processing power, storage, or bandwidth can take advantage of cloud computing services. However, this shift poses new challenges related to privacy and data security. Searchable Encryption (SE), which combines encryption and search techniques, addresses these issues (violation of data users' privacy) by allowing user data to be encrypted, transmitted to a cloud server, and searched using keywords. Despite its benefits, several recent real-world attacks have raised concerns about the security of searchable encryption. Ensuring forward and backward privacy is likely to become a standard requirement in the development of new SE systems. To address these issues, we propose a scheme that exclusively uses symmetric cryptographic primitives, achieving high communication efficiency and forward and backward privacy. In addition, we emphasize improved I/O efficiency because only the results of subsequent updates are loaded when searching. The time required to retrieve results is so significantly reduced compared to existing SE methods that we have shown that our scheme achieves superior efficiency. Moreover, by integrating blockchain network services with cloud services, we have developed a searchable intelligent cryptosystem suitable for lightweight smart devices. In our study conducted on the Ethereum network, we found our method to be both efficient and secure, especially when compared to methods such as PPSE and Jiang. The results indicate that our system delivers results in terms of performance and privacy within dynamic cloud environments making it a solution for protecting confidential information.

1. Introduction

The rapid development of cloud computing and mobile Internet is transforming people's lifestyles by aggregating substantial processing and storage power in the cloud. Service providers

*Corresponding author email : zaid.ameen@uobasrah.edu.iq

N: 1817-2695 (Print); 2411-524X (Online)
line at: https://jou.jobrs.edu.iq

offer on-demand services supporting a range of customized applications, benefiting customers with limited resources. Over the past decade, cloud computing has garnered significant interest from academia and industry due to its efficiency and low cost. However, the reliability of third-party service providers and the risks associated with storing unencrypted data are major concerns.

Searchable encryption has emerged as a solution to these privacy risks. Users who employ searchable symmetric encryption (SSE) systems can upload their encrypted files to a distant server and search them without the need for decryption. The concept of storage-as-a-service, offered by providers like Google Cloud, Azure Storage, and Amazon Cloud, has increased the demand for searchable encryption. This technique allows users to store data on distant servers while maintaining privacy. Traditional encryption protects outsourced data but is not effective for searching encrypted content. Searchable encryption addresses this by creating an encrypted index sent to the cloud provider along with the encrypted data. Clients can then use encrypted search tokens to query this index. The server uses these tokens and the encrypted index to find matches, ensuring data privacy even with an untrusted server.

Despite the development of various techniques [1], [2], [3], [4], [5] for searching encrypted data, some are not suitable for deployment on smart devices in an IoT cloud environment due to their lack of lightweight characteristics and insufficient privacy maintenance. Since 2012, several attacks [6] and [7] have emerged, allowing untrusted servers to recover keywords from clients' search tokens, thereby revealing significant information about the encrypted data. These attacks exploit information leakage that occurs during the search and update phases of searchable encryption schemes. For instance, the deterministic creation of search tokens can lead to a $search\ pattern$ [8] leakage, where the server identifies repeated search terms.

Effective Searchable Symmetric Encryption (SSE) systems [9], [10], [11], [12], [13], [14], [15] permit some information leakage, such as $access\ patterns$ [8]—the collection of documents containing the search term. Dynamic SSE techniques [16], [17] support update activities on the document collection. Zhang et al.'s [7] recent work demonstrated an adaptive attack that can fully disclose client requests by introducing a small number of files into the encrypted data storage. This attack's consequences are severe, as recovered keywords help the server decipher encrypted data and assist in further statistical attacks, underscoring the need for forward privacy.

Forward privacy ensures newly added files cannot be linked to prior search terms, preventing such attacks. Stefanov et al.[9] emphasized that a secure SSE system must provide both forward and backward privacy. Forward privacy prevents file-injection[7] attacks, while backward privacy [18] ensures that deleted files remain private. Initial forward privacy schemes were introduced in earlier research [19], with subsequent studies [10], [20], [21], [22] continuing to address this issue. Backward privacy schemes aim to minimize leakage that occurs when a search query for a keyword can be linked to deleted documents.

Emerging technologies like blockchain have facilitated frameworks allowing individuals to manage access to valuable private data, such as genetic and medical information. Numerous studies have explored using blockchain as an access control system, empowering individuals to control their health data. Recent research proposed methods for recording medical data access logs on blockchain's immutable ledger [23], [24], ensuring transparency and accountability. One significant challenge in healthcare is the inability of facilities to share data storage units, delaying patient treatment programs. Recent research [25] has developed token-based access mechanisms using blockchain smart contracts to address this, enabling efficient interoperability among health institutions and facilitating timely patient medical records exchange. This advancement aims to improve care coordination and expedite treatment processes.

Contributions: In this work, we develop and implement a single-keyword Searchable Symmetric Encryption (SSE) dynamic technique that uses two rounds for search queries, providing both forward privacy and Type-II backward privacy. This approach ensures that the system remains secure against adaptive attacks while maintaining user privacy during file additions and deletions. Our approach offers the following attributes:

1. **Forward Privacy**: To prevent the server from connecting later update tokens to earlier search tokens, new search tokens are created using a fresh key st.

2. **Backward Privacy**: The server is not informed of update operations (add or delete) while processing update queries, ensuring privacy during these operations. Our system achieves level two of backward privacy.
3. **Constant client storage**: Our proposed work is the DSSE framework that satisfies Smart Device Client SDC requirements while being effective, safe, and having a constant storage cost on the client side. Furthermore, in terms of processing complexity throughout the search and update procedures, our strategy is both practically and conceptually close to ideal. Nonetheless, Table 1 illustrates how well it compares to earlier methods.
4. **Efficient Index Management**: During a search, accessed items in the encrypted index are identified and removed, preventing the index from growing in size unnecessarily.
5. **Optimized Input/Output**: Previous search results, which are already compromised, are stored on the server in plaintext. This avoids the overhead of re-encrypting them, allowing for efficient and continuous reading of plaintext results at the optimal location.

The structure of the remainder of the paper is as follows: The most relevant SSE schemes are reviewed in Section 2. Section 3 provides the background and preliminary information necessary to understand our proposed plan. Section 4 outlines our proposed framework for secure data exchange, including implementation details. Section 5 presents the security evaluation, experimental setup, and performance analysis. Finally, Section 6 concludes the paper.

2. Previews work

Searchable Symmetric Encryption (SSE) and construction with linear search time were first presented by Song et al. [26]. Later on, Curtmola et al. enhanced this by offering the first design with sub-linear search time by tracking a list of document IDs for each keyword using an inverted index [8]. The static scenario, in which the encrypted database is kept on the cloud server without any further changes, has also been an issue of later studies [27], [28], [29].

To facilitate database modifications, Dynamic Searchable Symmetric Encryption ($DSSE$) was created to allow users to add and delete keyword/document combinations in the database [19], [30]. Increasing the computational complexity of the search protocol has been one of the main areas of interest for many academics. The first DSSE method with sub-linear search time was presented by Kamara et al. [31], although their solution revealed the hashes of the modified texts' keywords. Later, Kamara and Papamanthou raised the space complexity in order to solve this problem [16]. Nevertheless, these techniques are still susceptible to more sophisticated assaults including leakage-abuse attacks [6] and file injection attacks [7]. These assaults draw attention to the significance of forward security, which was explicitly established in [9] as an essential part of $DSSE$.

Despite Stefanov et al. [9] presented a forward-secure DSSE technique; their method has minimal computational complexity since it rebuilds the data structure at the update protocol level. Some systems, like Oblivious RAM[32], provide forward security using sophisticated cryptography; however, these methods are computationally expensive[20], [33]. Forward-secured schemes with theoretically optimal computing complexity exist but have certain limitations[10], [18]. $Sophos$, for instance, creates search tokens using an inverted index and forwards them to the server for further searching without revealing the actual terms[10]. $Sophos$ establishes a link between the search tokens and update tokens using trapdoor permutation, offering better theoretical processing complexity than[33]. However, since the trapdoor permutation relies on public-key primitives, $Sophos$ is inefficient in practical use. An informal mention of backward security can be found in [9], followed by the formal definition and structures providing this attribute provided by Bost et al.[18] . The systems Janus and $Diana_{\mathrm{del}}$, as described in[18], encounter inefficiencies with puncturable encryption and restrictions on keyword/document pairings, respectively. Subsequently, Chamani et al. [34] presented three strategies. Among them, Mitra requires two-round communication but achieves optimal computational and communication complexity. Conversely, the other two strategies trade off compute and transmission costs for backward security.

Our proposed DSSE framework addresses the weaknesses identified in previous works by ensuring both forward and backward privacy through the use of symmetric key operations, thereby enhancing performance and security. The system eliminates the need for public key operations, reducing computational overhead and making it suitable for use in resource-constrained environments. Additionally, our approach maintains constant client storage costs, which is a significant improvement over previous methods that often require substantial storage for maintaining encrypted indices and states.
Utilizing a three-party model comprising the data owner, the server with the encrypted database, and blockchain nodes housing local databases it lessens the computational load and storage cost on the client, our system guarantees data integrity and security. By generating different encrypted values for the same keyword with each update query. This architecture renders our framework suitable for various sensitive systems, thereby amplifying its practical value and widening its potential influence.

Table 1. compares many different DSSE schemes.

Scheme	Computation cost		Forward Security	Backward Security	Owner Storage
	Search	Update			
TWORAM[20]	$\hat{O}(a_w \log \mathrm{N} + \log^3 \mathrm{N})$	$\hat{O}(\log^2 \mathrm{N})$	×	×	$O(1)$
Mitra[34]	$O(a_w)$	$O(1)$	✓	✓	$O(m)$
SD_a[35]	$O(a_w + logN)$	$O(logN)$	✓	✓	$O(m)$
Sophos[18]	$O(a_w)$	$O(1)$	✓	×	$O(m)$
CLOSE-FB[36]	$O(a_w + Con)$	$O(Con)$	✓	✓	$O(1)$
Jiang et al.[37]	$O(s_w \times n_w \times o_w \times m)$	$O(logN)$	✓	×	$O(1)$
PPSE[38]	$O(s_w \times n_w \times o_w)$	$O(logN)$	✓	✓	$O(1)$
Our scheme	$O(o'_w)$	$O(1)$	✓	✓	$O(1)$

The number of (keyword, identifier) mappings is denoted by N. The variable m represents the number of distinct keywords w. Addition operations on a keyword w are denoted by a_w, while d_w signifies the number of delete operations performed on w. The total number of updates on w is represented by o_w, where $o_w = a_w + d_w$. Since the last search, the number of updates is indicated by o'_w , and n_w refers to the number of documents currently associated with w . The term $\hat{O}$ encapsulates the log logN components. Satisfied is indicated by ✓, and unsatisfied by ×. As of the last search, the constant Con represents the total number of computations that can be performed on the hash function. s_w reflects the count of searches executed for keyword w

3. Preliminaries

To understand our proposed framework, this section covers background information on blockchain, fully homomorphic encryption, cloud server architecture, and inverted index. The key concepts and notations used in this study are listed in the notation subsection.

3.1. Notation

In our system, λ serves as the indicator for the security parameter, while F is employed as a pseudorandom function. G functions as a pseudorandom permutation, with G^{-1} representing its inverse of pseudorandom permutation. The hash functions H_1, H_2 and H_3 are utilized, each tailored to produce outputs suitable for λ. The CSP, It is an abbreviation for Cloud Server Provider, plays a crucial role in our architecture. Additionally, the concatenation of strings a and b is denoted by $a||b$. Within our system, Doc refers to files containing a collection of keywords. $\boldsymbol{P}(O; B; S)$ signifies a protocol engaging three key participants: the data owner, blockchain, and server. $DSSE$, It is an

abbreviation for Dynamic Searchable Symmetric Encryption, is integral to our framework, and the XOR operation, represented by $\oplus$, forms a fundamental part of our cryptographic processes.

3.2. Blockchain

Blockchain, a distributed ledger of immutable and secure transactions, resides on nodes within a decentralized peer-to-peer (P2P) network. Nodes employ a consensus mechanism to validate a block containing all network transactions, which is compiled and appended to the blockchain[39]. This verified block is then linked to the most recent block using cryptographic hash pointers. A block consists of two main parts: the block header, containing metadata such as timestamp and hash values of previous and current blocks, and the block content, housing transaction data. There are three types of blockchain networks: consortium, private, and public, which find applications in various industries like healthcare and banking. These networks allow participants to conduct business without necessitating mutual trust. Smart contracts (SC) enforce the terms and conditions of agreements between parties. A smart contract is an immutable computer program recorded on a blockchain, automatically executing according to the predefined rules of a multiparty agreement. In this work, to enforce agreements between Data Owners (DOs) and Data Users (DUs), we establish a private blockchain network and deploy smart contracts on it.

Blockchain technology is crucial in enhancing the robustness of our DSSE framework. It eliminates the need for trust by creating a secure, immutable record of all transactions. Despite these advantages, trust in the underlying system and the smart contract code remains essential. Vigilance is needed to detect and address any bugs or vulnerabilities that could pose security risks.

Advantages of Our Framework:

- **Enhanced Security:** Immutable records on the blockchain significantly boost security.
- **Efficient Data Management:** Automated security policies via smart contracts streamline data management.
- **Decentralized Storage:** Data is distributed across multiple nodes, ensuring reliability even during failures.

Incorporating blockchain into our DSSE framework results in a secure, efficient, and resilient system.

3.3. System model

As shown in Figure 1, the parties in our system are the Storage Provider (CSP), Data Owner(DO), Data User(DU), and Blockchain Nodes(BN). The characteristics and roles of each party are shown as follows.

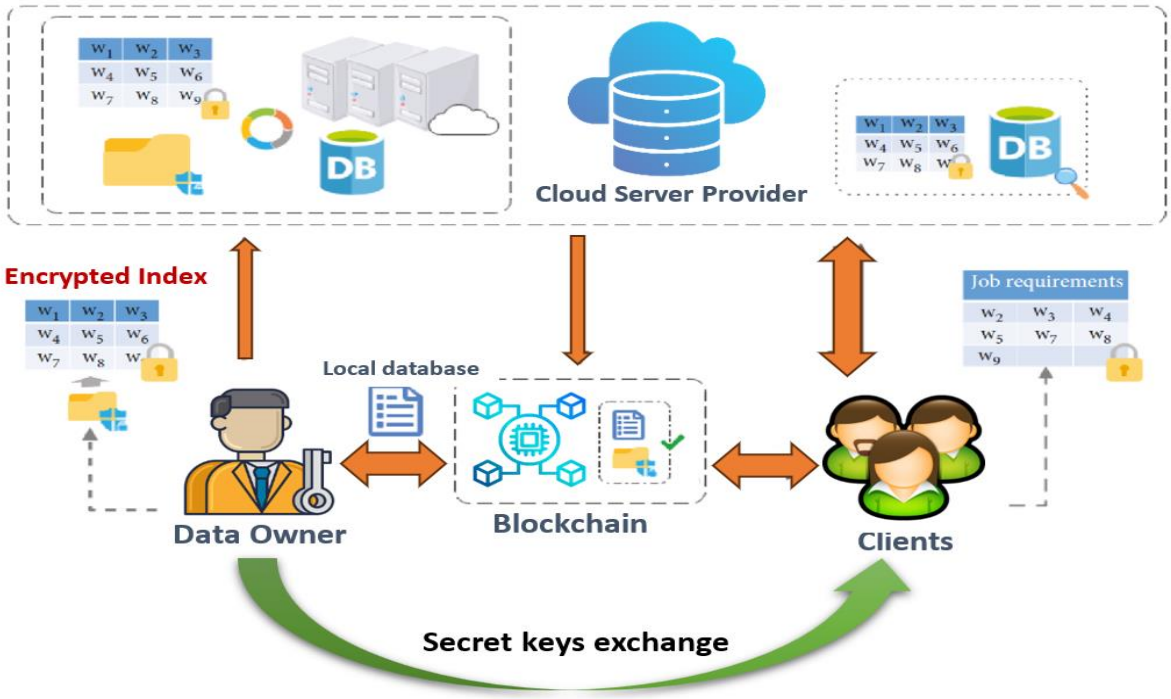

Fig. 1. Our Framework Model

a) Service providers with robust processing and storage capacities are known as Cloud Storage Providers (CSP). When a Data User (DU) makes a retrieval request, the CSP extracts the corresponding ciphertext from the encrypted data stored on behalf of the data sender (users). The final step involves returning the retrieval result to the DU and recording it on the blockchain.
b) The Data User (DU) generates a search token based on the keywords and the local database stored on the blockchain and then sends the trapdoor to the Cloud Storage Provider (CSP).
c) The blockchain network consists of nodes represented by candidates, hiring agencies, and other entities. Its primary responsibility is to maintain the network and enable smart contracts, which can be used to store user data and local databases.
d) Data Owners (DO) are responsible for maintaining the system and resolving issues. They play a pivotal role in allocating and distributing access tokens to users. Their tasks include generating encryption keys, creating and modifying the encrypted index, and generating keys for users upon request, especially when users initiate search operations. Additionally, they create local databases and store them on blockchains.

3.4. leakage Pattern

Let Q be a set of q-queries, with each pair (t, w)representing a keyword and t representing the query's timestamp. The following outlines the leaking of the following [40]:

1. ***Access Pattern***: This indicates the content of a document based on the query keyword. For each query keyword w, the access pattern is defined as $ap(w) = ID(w), where\ ID(w)$ is the document's ID number.

2. ***Search Pattern***: This shows the search patterns for each keyword w, defined as $sp(w) = \{t \mid (t, w) \in Q\}$.

3.5 The ***definition*** of our system consists of six probabilistic polynomial-time (PPT) algorithms, which are as follows: $(Setup, LocalRetrieve, Trapdoor, Search, Dec, and\ Update)$. Each of these algorithms plays a crucial role in the functionality and security of our scheme. The formal constructions of these algorithms are outlined as follows:

1. $(MSK; K_{count}, S_{count};\ P_{res}\ , E_{idx}\ ) \leftarrow \boldsymbol{Setup}(\lambda, \perp;\ \perp; \perp)$: This algorithm initializes the system parameters λ and generates the necessary cryptographic keys MSK. It sets up the environment for secure storage and retrieval operations. The algorithm generates a set of maps, which are subsequently stored on both the server and the blockchain.

2. $(K_{count}[w], S_{count}[w]) \leftarrow \boldsymbol{LocalRetrieve}(b_w)$: This algorithm is responsible for retrieving the current state from the local database stored on the blockchain. It ensures that the most recent information is accessed for processing subsequent operations. They are utilized when making updates and search requests.

3. $(K_{count}[w], S_{count}[w], b_w\ ) \leftarrow \boldsymbol{Trapdoor}(MSK, \mathrm{w})$: This algorithm generates the search token, also referred to as the trapdoor. The inputs for this process are the keyword to be searched and the master secret key (MSK). The output is the Search Token, which encapsulates the state of the most recent encrypted value along with the word counter. Cryptographic techniques are employed to ensure the keyword remains confidential while enabling the search functionality. These tokens are used when initiating a search request.

4. $E_{res} \leftarrow \boldsymbol{Search}(K_{count}[w], S_{count}[w], b_w\ )$: The inputs for this algorithm include the search token provided by the data owner. Represented by the current status $S_{count}[w]$ with the update counter $K_{count}[w]$ on the keyword. w The algorithm interacts with the server to locate the encrypted data associated with the keyword w. It ensures that the search operation is both efficient and secure, resulting in a set of encrypted values E_{res} as the output.

5. $P_{res} \leftarrow \boldsymbol{Dec}(E_{res})$: This decryption algorithm allows the data owner to decrypt the retrieved encrypted E_{res} results. It ensures that only authorized users can access the plaintext P_{res} data from the encrypted storage. In this step, the owner makes a filter for deletion value. They are utilized when initiating a search request.

6. $(MSK; K_{count}{}', S_{count}{}';\ P_{res}{}'\ , E_{idx}{}'\ ) \leftarrow \boldsymbol{Update}(MSK, \mathrm{ind}, \mathrm{doc}, \mathrm{op}; K_{count}, S_{count}; \mathrm{E}_{idx})$: This algorithm handles the addition and deletion operation *op* of keyword-document pairs in the encrypted index. It maintains the integrity and confidentiality of the data while allowing dynamic updates to the system. The output of this algorithm is modified for these maps $K_{count}{}', S_{count}{}';\ P_{res}{}'$ and $E_{idx}{}'$

These six algorithms collectively define the functionality and security properties of our dynamic searchable encryption scheme, providing a comprehensive framework for secure and efficient data storage and retrieval.

4. Our Optimize DSSE framework structure Overview

In this section, we introduce our system, which is the first method for searchable encryption that guarantees both forward and backward privacy. The primary objective of this scheme is to maintain optimal communication complexity while eliminating the need for public key operations. Unlike traditional approaches that pseudorandomly generate all states with a fixed key, our system employs a novel method where the data owner selects a random ephemeral key each time a new state is generated. In our approach, the current state for a keyword is derived by encrypting the previous state using the ephemeral key. This ephemeral key is then encrypted with the current state and stored on the server side. The state is integrated with the encrypted value contained within the encrypted index. The server can retrieve the ephemeral key only when it is aware of the current state. By using the ephemeral key, the server can decode the current state to obtain the previous state, enabling it to conduct searches by iteratively collecting all states. A key advantage of our system is that the client only needs to provide the server with the most recent state, keeping the

search token size constant. Forward privacy is ensured because the server cannot deduce unknown states from the currently known states and keys. Additionally, backward privacy is maintained as the server cannot distinguish previously added or deleted keywords, since the results are returned in encrypted form. The procedures of the scheme are depicted in Figure2. The algorithms that illustrate the forward and backward evolution of states in the update and search protocols are also provided below:

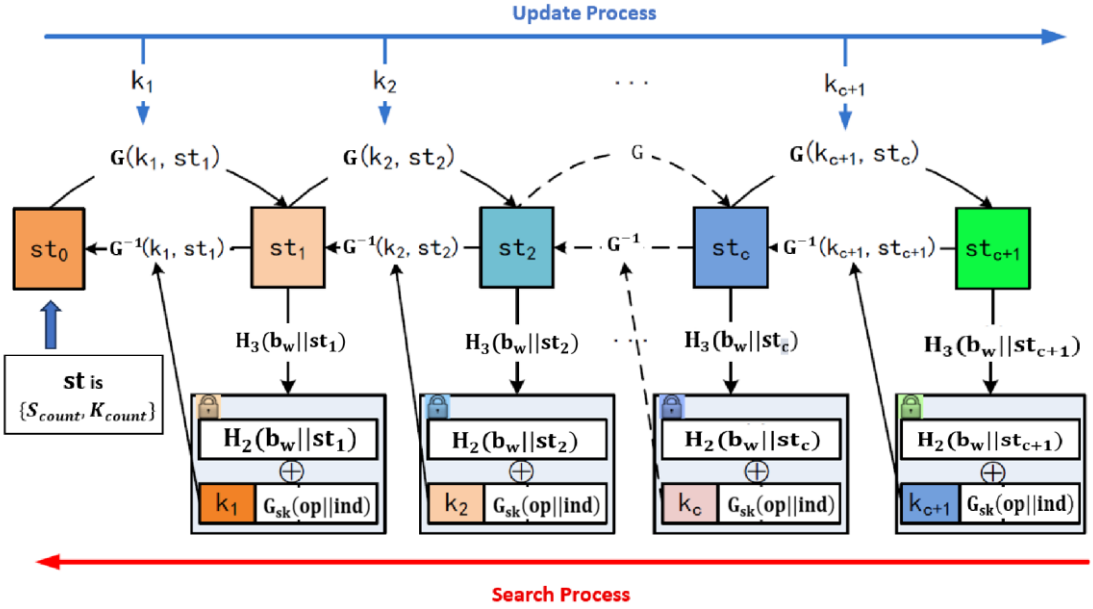

Fig. 2. The query for the keyword $\boldsymbol{w}$ from our system

This innovative design ensures secure, efficient, and privacy-preserving searchable encryption, making it a robust solution for protecting sensitive data in dynamic environments.

Setup protocol: In the setup protocol, the data owner generates MSK, K_{count}, and S_{count} according to the specifications outlined in the setup method. MSK represents a λ -bit long-term key designated for encrypting keywords. K_{count} and S_{count} contain empty maps representing local databases, including search counters and distinct keyword counters, which will record states on the blockchain network side. The inclusion of the long-term key ks prevents the server from autonomously generating tokens. Additionally, the data owner initializes two empty maps: P_{res}, intended for storing plaintext search results ($access\ pattern$), and E_{idx}, which holds the encrypted index. These maps are stored on the server.

$\boldsymbol{Setup}(\lambda, \perp; \perp; \perp)$
Data Owner:

1: $MSK \xleftarrow{\$} \{0,1\}^{\lambda}$ // store in owner

2: $K_{count} \leftarrow empty$ // send to the blockchain
$S_{count} \leftarrow empty$ // send to the blockchain

3: $E_{idx} \leftarrow empty\ map$ // send to server
$P_{res} \leftarrow empty\ map$ // send to server

Update data: When updating a file that contains a keyword ww with identifier ind, the data owner follows a systematic process to ensure both efficiency and security. Initially, the data owner retrieves the previous state st_c from the local state store on the blockchain side using the mappings (S$\leftarrow S_{count}[b_{w_i}]$, C$\leftarrow K_{count}[b_{w_i}]$) (lines 1-3 in the Update algorithm). Next, the data owner generates a random ephemeral key $K_C \xleftarrow{\$} \{0,1\}^{\lambda}$ and advances the state to the current state S_C using a pseudorandom permutation (lines 7-9). The local data retrieval location

b_{w_i} is generated using the private secret key MSK and the hash function H_1, following the function $b_{w_i} \leftarrow \mathrm{F}(MSK, \mathrm{H}_1(\mathrm{w}))$ (line 2). To maintain backward privacy, a secret key $sk_C \leftarrow F(MSK, w||C)$ is renewed with each update query. This renewal obscures the contents of the encrypted value, preventing the server from detecting additions or deletions during the operation op (line 9). The ephemeral key S_C is not stored on the owner or blockchain side but is embedded in the encrypted index entry E_{valu} that will be stored on the server side (line 10). The owner also generates a reference loc from the current state and the keyword (line 11). The pair (loc and E_{valu}) is sent to the server, which then updates its map E_{idx} accordingly (line 12). Finally, the local maps (K_{count} and S_{count}) on the blockchain side are updated to reflect these changes (line 13). This meticulous process ensures the integrity and privacy of the data during updates.

Update$(MSK, \mathrm{ind}, \mathrm{doc}, \mathrm{op}; K_{count}, S_{count}; \mathrm{E}_{idx})$
Data Owner:
1: $for\ i\ =\ 1\ to\ |doc|$
2: $b_{w_i} \leftarrow \mathrm{F}(MSK, \mathrm{H}_1(\mathrm{w}))$
Retrieve from the blockchain network
3: S$\leftarrow S_{count}[b_{w_i}]$, C$\leftarrow K_{count}[b_{w_i}]$
4: **if** $(S, K) = \perp$ then
5: C$\leftarrow$ 0, S$\overset{\$}{\leftarrow} \{0,1\}^{\lambda}$
6: C$\leftarrow C + 1$
7: $K_C \overset{\$}{\leftarrow} \{0,1\}^{\lambda}$
8: $S_C \leftarrow G(K_C, S_{C-1}||C\ )$
9: $sk_C \leftarrow F(MSK, w||C)$
10: $E_{valu} \leftarrow (G(\ sk_C\ , (op\ \|ind))||\ K_C) \oplus H_2(b_{w_i}||S_C||C)$
11: $loc \leftarrow H_3(b_{w_i}||S_C||C)$
12: Update $E_{idx}[loc] \leftarrow E_{valu}$ in server
Update $K_{count}[\ b_{w_i}] \leftarrow\ C$ in blockchain
13: Update $S_{count}\ [\ b_{w_i}] \leftarrow\ S$ in blockchain

Search Data: To search a keyword w, the data owner follows a systematic algorithm. Initially, the data owner retrieves the current state S and C from the local database stored on the blockchain, denoted as S$\leftarrow S_{count}[b_{w_i}]$, $C \leftarrow K_{count}[b_{w_i}]$ (lines 1-2 in the search algorithm). The first step involves verifying whether the keyword w exists in the encrypted index by examining the retrieved values from the blockchain. If the keyword is not found in the encrypted index, the process terminates (lines 3-4). Subsequently, the algorithm computes the search trapdoor and sends a search token, which includes the encrypted keyword and the current state (S , C , b_w), to the server (lines 5-6). Based on the current state S, the server generates all previous states S_{i-1} and identifies the corresponding update sequence. Within the algorithm's for-loop, the server recovers the ephemeral key K_i (line 10), which is then utilized to retrieve the previous state $S_{i-1} \leftarrow G^{-1}(K_i, S_i||i\ )$ (line 13). The server conducts a backward search through the update sequence using the state S_i. The encrypted values associated with the keyword are compiled into a set E_{res} (line 12). To mitigate access pattern leakage and optimize storage efficiency, these results are deleted from the server after extraction (line 11).
Upon retrieving all encrypted values associated with the keyword and updating the encrypted index, the encrypted results E_{res} are transmitted to the data owner. Given that both "add" and "delete" operations are permissible, the data owner must ensure that deleted files are excluded from the result set. This is achieved by maintaining a set P_{res}, which contains the identifiers of deleted files throughout the search process. When a delete update is encountered, the server inserts the file identifier ind into P_{res}. Conversely, if an add update is detected and ind is present in P_{res}, the data owner removes ind from P_{res}. This meticulous filtering process

ensures that only the added results are retained. The data owner performs this filtering to ensure the server remains unaware of which data was subsequently deleted or added, thus maintaining backward privacy (lines 14-21). After completing the search, the data owner updates the local data for the keyword to a null value ⊥ on the blockchain, ensuring the integrity and confidentiality of the search process.

$\boldsymbol{Search}(MSK, w, op; K_{count}, S_{count}; E_{idx}, P_{res})$
Data Owner:
1: $b_w \leftarrow \mathrm{F}(MSK, \mathrm{H}_1(\mathrm{w}))$
Retrieve from the blockchain network
2: S$\leftarrow S_{count}[b_{w_i}]$, C$\leftarrow K_{count}[b_{w_i}]$
3: **if** $(S, C) = \perp$ then
4: If the keyword w not inserted then return $\emptyset$;
5: ***else***
6: Send (S , C , b_w) to server
Server:
7: $E_{res} \leftarrow \{\}$
8: $for\ i\ =\ C\ to\ 1\ do$
9: $loc \leftarrow H_3(b_w||S_i||i)$
10: $(G(\ sk_C\ , (op\ \|ind))||\ K_i) \leftarrow E_{idx}[loc] \oplus H_2(b_w||S_i||i)$
11: Delete $E_{idx}[loc]$
12: $E_{res} \leftarrow E_{res} \cup G(\ sk_C\ , (op\ \|ind))$
13: $S_{i-1} \leftarrow G^{-1}(K_i, S_i||i\ )$
Send E_{res} to owner
Data Owner:
14: $P_{res} \leftarrow \{\}$
15: $for\ i\ =\ 1\ to\ |\ E_{res}\ |\ do$
16: $sk_i \leftarrow F(MSK, w||i)$
17: (op ‖ind) $\leftarrow G^{-1}(sk_i, E_{res})$
18: $\boldsymbol{if}\ op = "add"\ \ then$
19: $P_{res} \leftarrow\ P_{res} \cup ind$
20: else
21: $P_{res} \leftarrow\ P_{res} -\ ind$
Update $K_{count}[\ b_{w_i}] \leftarrow \perp$ in Blockchain
22: Update $S_{count}\ [\ b_{w_i}] \leftarrow \perp$ in Blockchain

5. Experimental analysis

A small Ethereum network was constructed locally to experiment with the distinct performance features and security of our proposed system. The only difference between the simulated network and the actual Ethereum environment is that the mining block time is set to 0. With this approach, we can concentrate on the smart contract's search functionality even amidst Ethereum's complex network environments and laborious mining procedures, which include broadcast and transaction mining delays. We assessed the search time by returning a predefined number of matching documents, primarily evaluating the effectiveness of the search and update phases. To determine the updating cost, we added and deleted a set number of files and measured the associated time and gas consumption. We compared our proposed system with existing DSSE schemes, specifically PPSE [38] and Jiang [37] approaches, which operate in similar blockchain environments. The hardware setup consists of an Intel (R) Core (TM) i5-10400F CPU @2.90 GHz processor with 8 GB of RAM, and the environment is based on a Windows 11 64-bit operating system, x64-based processor. Our proposed system utilizes the original dataset sourced from the Enron email[41] dataset, from which a subset is extracted for testing purposes. During the experiment, the smart contract is implemented using the Solidity language, while Python is employed as the language for interacting with the smart contract.

Additionally, we provide the $PRF\ F$ implemented with AES-128/256 and $PRP\ G$ implemented with AES-128/128. The utilization of stronger hash functions such as SHA-3 does not significantly impact the efficiency of our scheme, as hashing is not the primary cornerstone of our design. Our suggested scheme simulation experiment assesses the search time by returning a set number of matching documents, primarily testing the effectiveness of the search and updating phases. A set number of files are added and deleted to determine the updating cost. The PPSE[38], and Jiang[37] of the same backward private level and the same blockchain environment, which also stores indexes and documents independently, are compared with our suggested approach. It's important to note that the results for the comparison schemes were obtained from the article[38]. Our system mitigates side-channel attacks by using unique ephemeral keys for each state transition, ensuring timing patterns don't reveal encrypted data. It maintains consistent query response times, preventing attackers from inferring data size based on processing speed. Additionally, pseudorandom permutations in state transitions add further obfuscation, protecting against information leakage through observable patterns. This structure ensures a robust defense against side-channel vulnerabilities.

We assessed the search time by returning a predefined number of matching documents, primarily evaluating the effectiveness of the search and update phases. To determine the updating cost, we added and deleted a set number of files and measured the associated time and gas consumption. We compared our proposed system with existing DSSE schemes, specifically PPSE and Jiang's approach, which operate in similar blockchain environments. The following is an analysis of the outcomes.

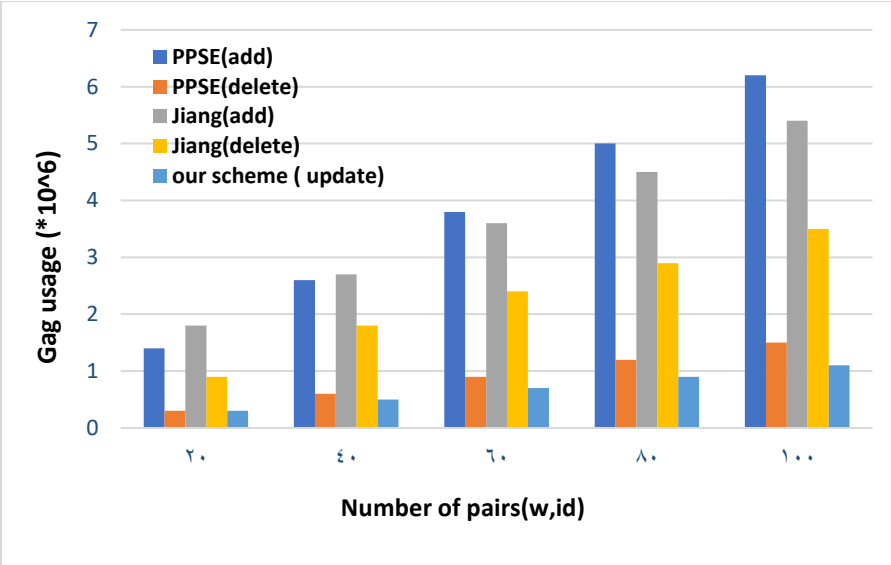

Fig.3. updates gas consumption

The number of matched documents is set from 100 to 500 for searching to demonstrate the basic method. The growing trend of search time as the number of matched documents varies is seen in Figure 5. After carrying out each strategy thirty times, we average the results. In our scheme, a comparison with others reveals that as the number of matched documents increases, there is a slight decrease in the efficiency of the search algorithm in terms of time cost. In contrast, for other schemes, the increase is linear and substantial with the rise in the number of retrieved documents. The amount of data to be retrieved and the duration of the search is increasing with the size of the dataset in the cloud environment. The loading time increases with the number, resulting in worse algorithm execution efficiency compared to the cloud environment. Figure 4, however, shows that our scheme's query efficiency is substantially quicker than that of the other schemes in the cloud environment. Our system is quicker than the asymmetric encryption employed in comparable schemes since it is based on symmetric primitive encryption technologies. Furthermore, compared to research on the identical blockchain environment, the security in our scheme is noticeably greater than [37]. Query

efficiency is an advantage. To protect data, search functions on the blockchain are carried out using smart contracts.

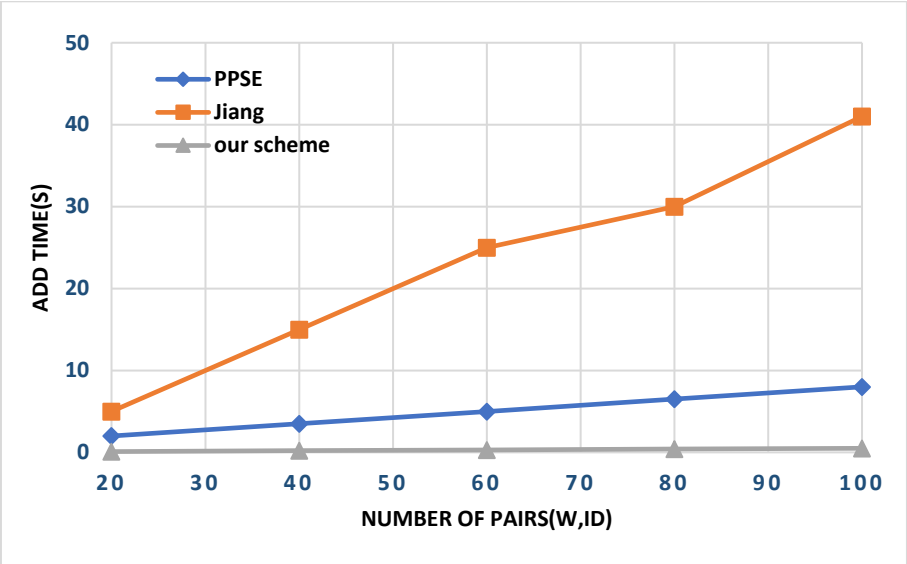

Fig. 4. Update addition time

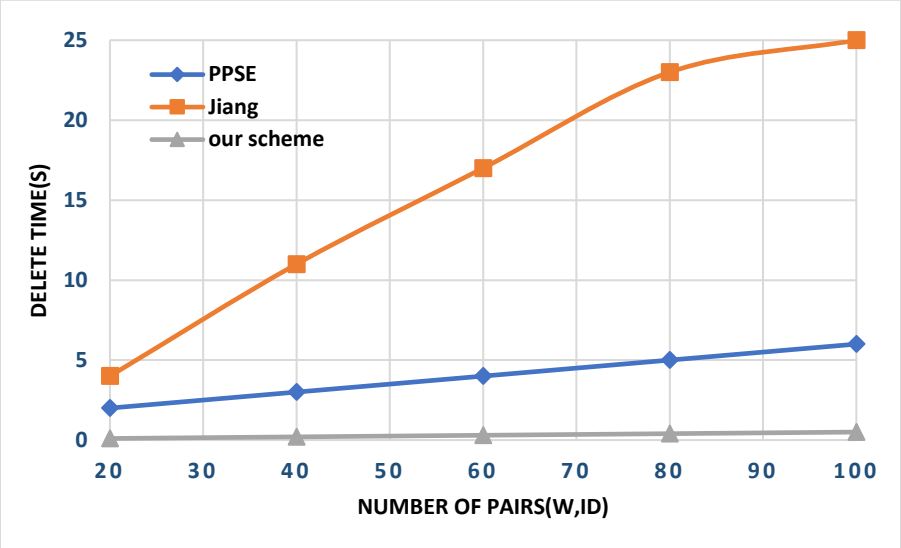

Fig. 5. Update delete time

The usage and time expenses that vary depending on how many files are added or removed are shown in Figures 3–5. The number of files added and removed causes a linear rise in both the execution time and gas consumption as in Figure 3 (gas is a unit of measurement used to quantify the amount of computing labor required to complete an action on the network blockchain). The (keyword, identifier) inverted index pairs in the file are raised from 20 to 100 for testing by choosing files of varying sizes. Figure 4 and Figure 5 demonstrate that, in the cloud context, the update efficiency of the Jiang [37] and PPSE[38] systems are much less than our method. In the same blockchain context, our system design not only offers clear performance advantages in updates but also achieves a higher level of backward privacy compared to Jiang [37], which does not ensure backward privacy. The graphs show that for comparison systems, the time growth is substantially linear as the result size increases, but in our system, the cost of time growth is much less due to changes in the database size. This is not surprising, as our system uses symmetric encryption to ensure forward and backward privacy, whereas Jiang [37] and PPSE [38] approaches rely on one-way trapdoor permutations (e.g.

implemented with RSA). All changes and deletions made to the blockchain are handled via smart contracts, which are inexpensive, decentralized, and have a high timeliness.
The experimental results demonstrate the effectiveness and efficiency of our proposed DSSE framework, particularly in terms of search and update operations. By leveraging symmetric encryption and optimizing storage efficiency, our system achieves superior performance without compromising security, making it a robust solution for protecting sensitive data in dynamic environments. Quickly retrieves data when searching and updating requests compared to other systems.

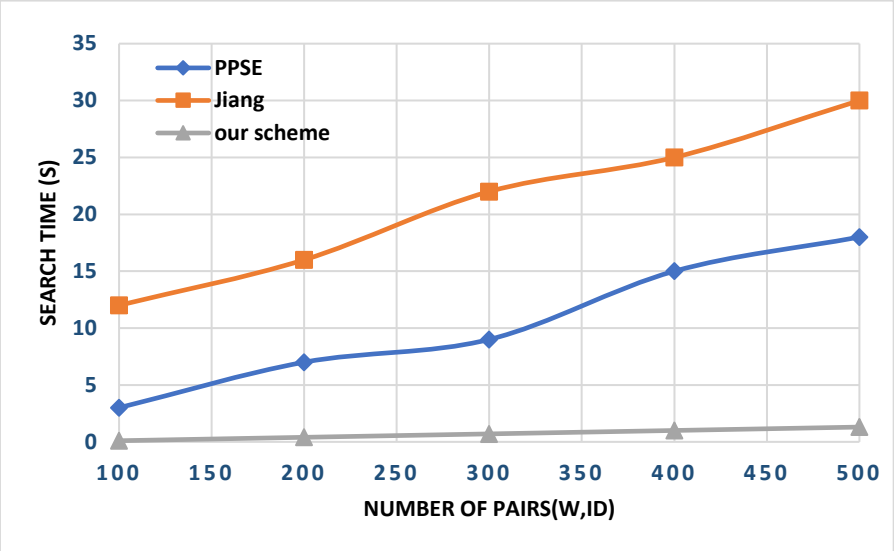

Fig. 6. Query search time.

In conclusion, the proposed scheme achieves the following features.
Security
DSSE framework provides strong security against attacks and ensures data privacy. It also includes forward and backward privacy to prevent the server from learning about the new or deleted documents. We secure secret keys by providing ephemeral keys and utilizing secure encryption ensuring data cannot be read by unauthorized parties.

Performance
Our scheme is tested for efficient performance on a local Ethereum network. The gas consumption for the updates is low, even though search times are far better than the other schemes. Note that this serves as an optimization under the hood (symmetric encryption is lighter than asymmetric).

Scalability
Technically, our DSSE scheme can manage large data sets well. Experimental results demonstrate that the execution time and gas consumption increase in linear form according to the number of files, guaranteeing performance efficiency.

Privacy
Our framework protects both data owner and client privacy. Data is stored encrypted, and pseudorandom permutations ensure the server cannot infer information. The search process maintains confidentiality, providing forward and backward privacy to safeguard data operations.

6. Summary

In this paper, we emphasize the practical application of Dynamic Searchable Symmetric Encryption (DSSE) and introduce the first efficient framework that maintains constant client storage Our suggestion is a novel dynamic SSE technique that addresses both kinds of privacy

without requiring complex asymmetric procedures. With this approach, we may surpass previous techniques! Our strategy operates in a three-party context. The data owner, who manages key management and data encryption, is the first party. Serving as a server, the second party has a large amount of storage and powerful processing to handle encrypted data. The third-party consists of blockchain nodes that manage local database storage, ensuring data integrity and security. To enhance efficiency, we propose to use an encrypted inverted index on the server to map keywords to their corresponding document IDs. This approach greatly reduces client-side calculations, limiting the server's role primarily to storage tasks rather than task processing. All calculations in our system are performed server-side.As a result, the cost of generating a search token, referred to as a trapdoor, remains constant regardless of the frequency of keyword updates. This design ensures that our system is compatible with smart devices, as there is no need for local data storage or intensive computations on these devices. Thus, our framwork is suitable for a wide range of service providers and can be adapted as a global model. This versatility makes our solution applicable across various critical systems, enhancing its practical utility and expanding its potential impact.

7. Future Work

While demonstrating that forward and backward privacy can be achieved with high efficiency is a positive step, further research is needed before SSE methods can be employed in scenarios where information leakage is undesirable. This ongoing work will help ensure the robustness and applicability of SSE in such sensitive contexts.

8. References

[1] M. A. Al Sibahee, A. I. Abdulsada, Z. A. Abduljabbar, J. Ma, V. O. Nyangaresi, and S. M. Umran, “Lightweight, Secure, Similar-Document Retrieval over Encrypted Data,” *Applied Sciences*, vol. 11, no. 24, p. 12040, 2021, Doi: https://doi.org/10.3390/app112412040

[2] Z. A. Abduljabbar *et al.*, “SEPIM: Secure and efficient private image matching,” *Applied Sciences*, vol. 6, no. 8, p. 213, 2016. Doi: https://doi.org/10.3390/app6080213

[3] M. A. Al Sibahee *et al.*, “Lightweight secure message delivery for E2E S2S communication in the IoT-cloud system,” *IEEE Access*, vol. 8, pp. 218331–218347, 2020, Doi: 10.1109/ACCESS.2020.3041809

[4] M. A. Al Sibahee *et al.*, “Efficient encrypted image retrieval in IoT-cloud with multi-user authentication,” *Int J Distrib Sens Netw*, vol. 14, no. 2, p. 1550147718761814, 2018, Accessed: Jun. 18, 2024. Doi: https://doi.org/10.1177/1550147718761814

[5] Z. A. Abduljabbar, A. Ibrahim, M. A. Hussain, Z. A. Hussien, M. A. Al Sibahee, and S. Lu, “EEIRI: Efficient encrypted image retrieval in IoT-cloud,” *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 13, no. 11, pp. 5692–5716, 2019, Accessed: Jun. 18, 2024. Doi: https://doi.org/10.3837/tiis.2019.11.023

[6] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, “Leakage-abuse attacks against searchable encryption,” in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 668–679. Accessed: Jun. 18, 2024. Doi: https://doi.org/10.1145/2810103.2813700

[7] Y. Zhang, J. Katz, and C. Papamanthou, “All your queries are belong to us: The power of file-injection attacks on searchable encryption,” *Cryptology ePrint Archive*, 2016.

[8] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions,” in *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 79–88. Accessed: Jun. 18, 2024. Doi: https://doi.org/10.1145/1180405.1180417

[9] E. Stefanov, C. Papamanthou, and E. Shi, “Practical dynamic searchable encryption with small leakage,” *Cryptology ePrint Archive*, 2013, Accessed: Jun. 18, 2024. Doi: https://ia.cr/2013/832

[10] R. Bost, "∑ οφος: Forward secure searchable encryption," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1143–1154. Accessed: Jun. 18, 2024. Doi: https://doi.org/10.1145/2976749.2978303
[11] D. Cash *et al.*, "Dynamic searchable encryption in very-large databases: Data structures and implementation," *Cryptology ePrint Archive*, 2014, Accessed: Jun. 18, 2024. Doi: 10.14722/ndss.2014.23264
[12] S. S. Bilbul and A. I. Abdulsada, "Backward Private Searchable Symmetric Encryption with Improved Locality.," *Iraqi Journal for Electrical & Electronic Engineering*, vol. 17, no. 2, 2021, Accessed: Jun. 18, 2024. Doi: 10.37917/ijeee.17.2.3
[13] S. S. Bulbul and A. I. Abdulsada, "Security proof for backward searchable encryption scheme," *Journal of Basrah Researches (Sciences)*, vol. 47, no. 1, 2021.
[14] S. S. Bulbul *et al.*, "A provably lightweight and secure DSSE scheme, with a constant storage cost for a smart device client," *PLoS One*, vol. 19, no. 4, p. e0301277, 2024, Doi: https://doi.org/10.1371/journal.pone.0301277
[15] S. S. Bulbul, Z. A. Abduljabbar, D. F. Najem, V. O. Nyangaresi, J. Ma, and A. J. Y. Aldarwish, "Fast Multi-User Searchable Encryption with Forward and Backward Private Access Control," *Journal of Sensor and Actuator Networks*, vol. 13, no. 1, p. 12, 2024, Doi: https://doi.org/10.3390/jsan13010012
[16] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *International conference on financial cryptography and data security*, Springer, 2013, pp. 258–274. Doi: 10.1007/978-3-642-39884-1_22
[17] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part III 36*, Springer, 2017, pp. 94–124. Doi: 10.1007/978-3-319-56617-7_4
[18] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1465–1482. Doi: https://doi.org/10.1145/3133956.3133980
[19] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *International conference on applied cryptography and network security*, Springer, 2005, pp. 442–455. Doi: 10.1007/11496137_30
[20] S. Garg, P. Mohassel, and C. Papamanthou, "TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption," in *Annual International Cryptology Conference*, Springer, 2016, pp. 563–592. Doi: 10.1007/978-3-662-53015-3_20
[21] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with optimized I/O efficiency," *IEEE Trans Dependable Secure Comput*, vol. 17, no. 5, pp. 912–927, 2018, Doi: 10.1109/TDSC.2018.2822294
[22] M. A. Al Sibahee, C. Luo, J. Zhang, Y. Huang, and Z. A. Abduljabbar, "Dynamic Searchable Scheme with Forward Privacy for Encrypted Document Similarity," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, IEEE, 2023, pp. 1653–1660. Doi: 10.1109/TrustCom60117.2023.00225
[23] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "Medblock: Efficient and secure medical data sharing via blockchain," *J Med Syst*, vol. 42, pp. 1–11, 2018, Doi: 10.1007/s10916-018-0993-7
[24] J. Sun, X. Yao, S. Wang, and Y. Wu, "Blockchain-based secure storage and access scheme for electronic medical records in IPFS," *IEEE access*, vol. 8, pp. 59389–59401, 2020, Doi: 10.1109/ACCESS.2020.2982964
[25] R. H. Hylock and X. Zeng, "A blockchain framework for patient-centered health records and exchange (HealthChain): evaluation and proof-of-concept study," *J Med Internet Res*, vol. 21, no. 8, p. e13592, 2019, Doi: 10.2196/13592

[26] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE symposium on security and privacy. S&P 2000*, IEEE, 2000, pp. 44–55. Doi:10.1109/SECPRI.2000.848445
[27] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *International conference on the theory and application of cryptology and information security*, Springer, 2010, pp. 577–594. Doi: 10.1007/978-3-642-17373-8_33
[28] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, Springer, 2013, pp. 353–373. Doi: 10.1007/978-3-642-40041-4_20
[29] A. Bossuat, R. Bost, P.-A. Fouque, B. Minaud, and M. Reichle, "SSE and SSD: page-efficient searchable symmetric encryption," in *Annual International Cryptology Conference*, Springer, 2021, pp. 157–184. Doi: 10.1007/978-3-030-84252-9_6
[30] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, IEEE, 2018, pp. 792–800. Doi: 10.1109/INFOCOM.2018.8485890
[31] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 965–976. Doi: 10.1145/2382196.2382298
[32] E. Stefanov *et al.*, "Path ORAM: an extremely simple oblivious RAM protocol," *Journal of the ACM (JACM)*, vol. 65, no. 4, pp. 1–26, 2018, Doi: 10.1145/2508859.2516660
[33] M. Naveed, "The fallacy of composition of oblivious ram and searchable encryption," *Cryptology ePrint Archive*, 2015, Doi: https://eprint.iacr.org/2015/668.pdf
[34] J. Ghareh Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1038–1055. Doi: 10.1145/3243734.3243833
[35] I. Demertzis, J. G. Chamani, D. Papadopoulos, and C. Papamanthou, "Dynamic searchable encryption with small client storage," *Cryptology ePrint Archive*, 2019, Doi: 10.14722/ndss.2020.24423
[36] K. He, J. Chen, Q. Zhou, R. Du, and Y. Xiang, "Secure dynamic searchable symmetric encryption with constant client storage cost," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1538–1549, 2020, Doi: 10.1109/TIFS.2020.3033412
[37] S. Jiang, J. Liu, L. Wang, and S.-M. Yoo, "Verifiable search meets blockchain: A privacy-preserving framework for outsourced encrypted data," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, 2019, pp. 1–6. Doi: 10.1109/ICC.2019.8761146
[38] R. Du, C. Ma, and M. Li, "Privacy-preserving searchable encryption scheme based on public and private blockchains," *Tsinghua Sci Technol*, vol. 28, no. 1, pp. 13–26, 2022, Doi: 10.26599/tst.2021.9010070
[39] E. J. De Aguiar, B. S. Faiçal, B. Krishnamachari, and J. Ueyama, "A survey of blockchain-based strategies for healthcare," *ACM Computing Surveys (CSUR)*, vol. 53, no. 2, pp. 1–27, 2020, Doi: 10.1145/3376915
[40] Q. Song, Z. Liu, J. Cao, K. Sun, Q. Li, and C. Wang, "SAP-SSE: Protecting search patterns and access patterns in searchable symmetric encryption," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1795–1809, 2020, Doi: 10.1109/TIFS.2020.3042058
[41] Enron company, "Email Dataset." Doi: https://www.cs.cmu.edu/~enron/, Accessed: Jun. 21, 2024

Journal of Basrah Researches (Sciences) **50**(1), 304 (2024)
DOI: https://doi.org/10.56714/bjrs.50.1.24

نموذج الملخص باللغة العربية لمجلة ابحاث البصرة (العلميات)

سالم صباح بلبل [1]، زيد أمين عبد الجبار [2،*]

[1]المديرية العامة للتربية، البصرة، وزارة التربية، البصرة، 61004، العراق

[2]قسم علوم الحاسوب، كلية التربية للعلوم الصرفة، جامعة البصرة، البصرة، 61004، العراق

معلومات البحث

الاستلام 16 ايار 2024
القبول 25 حزيران 2024
النشر 30 حزيران 2024

الكلمات المفتاحية

كفاءة الإدخال/الإخراج، شبكة blockchain، الأدوات البدائية المتماثلة، التشفير القابل للبحث (SE)، الأمان الخلفي، الأمان الأمامي.

Citation: Salim S. B. , Zaid A. A., J. Basrah Res. (Sci.) 50(1), 304 (2024).
DOI:https://doi.org/10.56714/bjrs.50.1.24

الملخص

يمكن لأصحاب البيانات الذين يسعون إلى تحسين قوة المعالجة أو التخزين أو النطاق الترددي الاستفادة من خدمات الحوسبة السحابية. ومع ذلك، فإن هذا التحول يطرح تحديات جديدة تتعلق بالخصوصية وأمن البيانات. يعالج التشفير القابل للبحث (SE)، الذي يجمع بين تقنيات التشفير والبحث، هذه القضايا (انتهاك خصوصية مستخدمي البيانات) من خلال السماح بتشفير بيانات المستخدم ونقلها إلى خادم سحابي والبحث باستخدام الكلمات الرئيسية. على الرغم من فوائده، أثارت العديد من الهجمات الواقعية الأخيرة مخاوف بشأن أمان التشفير القابل للبحث. من المرجح أن يصبح ضمان الخصوصية الأمامية والخلفية متطلبًا قياسيًا في تطوير أنظمة SE الجديدة. لمعالجة هذه القضايا، نقترح نظامًا يستخدم حصريًا البدائيات التشفيرية المتماثلة، مما يحقق كفاءة اتصال عالية وخصوصية أمامية وخلفية. بالإضافة إلى ذلك، نؤكد على كفاءة الإدخال / الإخراج المحسنة لأنه يتم تحميل نتائج التحديثات اللاحقة فقط عند البحث. يتم تقليل الوقت المطلوب لاسترداد النتائج بشكل كبير مقارنة بأساليب SE الحالية التي أظهرنا أن مخططنا يحقق كفاءة فائقة. علاوة على ذلك، من خلال دمج خدمات شبكة blockchain مع الخدمات السحابية، قمنا بتطوير نظام تشفير ذكي قابل للبحث مناسب للأجهزة الذكية خفيفة الوزن. في دراستنا التي أجريناها على شبكة Ethereum، وجدنا أن طريقتنا فعّالة وآمنة، خاصة عند مقارنتها بأساليب مثل PPSE وJiang. تشير النتائج إلى أن نظامنا يحقق نتائج من حيث الأداء والخصوصية في بيئات سحابية ديناميكية، مما يجعله حلاً لحماية المعلومات السرية.

***Corresponding author email :** zaid.ameen@uobasrah.edu.iq

N: 1817-2695 (Print); 2411-524X (Online)
line at: https://jou.jobrs.edu.iq