Enabling Secure Recovery of Deleted Data Indexes via Searchable Symmetric Encryption with Backward Privacy

Menglei Wang[1], Haowen Tan[2,3,4,5,6], Wenying Zheng[7], Haoyu Ma[8,9]*

[1] School of Computer Science, Nanjing University of Information Science & Technology, China
[2] School of Information Science and Engineering (School of Cyber Science and Technology), Zhejiang Sci-Tech University, China
[3] Zhejiang Provincial Key Laboratory of Digital Fashion and Data Governance, Zhejiang Sci-Tech University, China
[4] Zhejiang Provincial International Cooperation Base for Science and Technology on Cloud Computing Security and Data Aggregation, Zhejiang Sci-Tech University, China
[5] Zhejiang Provincial Innovation Center of Advanced Textile Technology, Shaoxing, China
[6] Guangdong Key Laboratory of Blockchain Security, Guangzhou University, China
[7] School of Computer Science and Technology, Zhejiang Sci-Tech University, China
[8] School of Cyberspace Science and Technology, Beijing Jiaotong University, China
[9] Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, China
mengleiw2000@126.com, hwtan@zstu.edu.cn, zhengwy0501@126.com, hyma1@bjtu.edu.cn

Abstract

Searchable encryption has received considerable attention due to its ability to efficiently search encrypted private data while ensuring privacy protection. As the importance of robustness in encryption schemes continues to grow, research has increasingly focused on finding a balance between security, robustness, and performance. Robustness is critical for handling repetitive or frequent update operations, but there is a significant gap in addressing potential issues caused by data users, such as administrators. Current approaches largely focus on defending against external attackers and ensuring system stability during repetitive operations, but they fail to address how to reverse legitimate yet unintended update queries. In order to improve the practicality of the scheme, we present the first investigation into searchable encryption schemes with reversible properties. Our proposed scheme designs a robust and reversible data structure, achieved through the use of key-updatable pseudorandom functions, which can be applied to any collection of data for which an index can be constructed. Through a combination of rigorous theoretical analysis and comprehensive experimental evaluation, we demonstrate that our scheme successfully balances security, reversibility, and performance without compromising efficiency.

Keywords: Searchable symmetric encryption, Robust, Rollback, Privacy protection

1 Introduction

In today's rapidly advancing cloud computing landscape, data outsourcing has become a critical concern as users increasingly prioritize privacy. While data encryption is commonly used to mitigate privacy risks, it also introduces new challenges [1] in enabling data search and ensuring data availability post-encryption [2-3]. Within this context, searchable encryption offers a promising approach to balance security with data accessibility.

Current research in searchable encryption primarily focuses on enhancing search functionality and exploring applications across various fields [4-6]. However, these efforts largely overlook the need for robust systems. For instance, Xu et al. [7] introduced robustness by employing pseudo-random functions to create adaptive index structures, mainly achieved through delete operations that reduce index redundancy. Yet, a key limitation of this approach is that erroneous deletions can cause irreversible loss of the entire index and its content. To address this gap, we propose a searchable encryption scheme with reversibility to support data recovery. Our approach also introduces unique indexing mechanisms that support both robustness and reversibility.

As digital technology evolves, the value of data continues to increase, such as sensitive medical data, which underscores the urgency of taking effective protection measures. While encryption is essential for protecting sensitive data, traditional methods often struggle to cope with disorganized ciphertext structures, limiting the efficiency of data indexing and retrieval. Searchable encryption has emerged as an important solution that not only supports secure data search, but also safeguards data confidentiality. However, most of the existing solutions focus on external attacks, while ignoring the potential misuse caused by internal negligence. Misuse through negligence. For sensitive data, such irreversible errors can have serious consequences. Serious consequences. To address the shortcomings of existing schemes, we investigate how to achieve query reversibility while ensuring security and robustness, so that the index can be restored to its pre-update state after an update operation.

*Corresponding Author: Wenying Zheng; Email: zhengwy0501@126.com
DOI: https://doi.org/10.70003/160792642026112706003

We investigate how to achieve query reversibility while guaranteeing security and robustness. To better understand reversibility, we use healthcare data as an example and extend the data set to any dataset that can be indexed.

1.1 Related Work

Since the discovery of searchable encryption in 2000, encryption schemes with more rich feature-rich encryption have appeared, e.g., [8-11], and the integration with practice has evolved, e.g., [12-14]. Later, Kamara et al [15] added dynamic updates to searchable encryption, and since then, dynamic searchable encryption (DSSE) has become a research hotspot, and as a result, DSSE has become a common cryptographic tool combined with practical applications, e.g., [16-19]. Among the technologies used in many schemes, blockchain technology accounts for a relatively high proportion. However, the blockchain itself is a public, decentralized distributed ledger where all data and transactions are public. This setup can only ensure privacy under the existing security model, not necessarily under the new security model. In addition, blockchain possesses other characteristics [20] that can be integrated with searchable encryption, which will not be further discussed here. Therefore to ensure better protection of data privacy, adding permission control to searchable encryption has become a new research direction, e.g., [21]. However, these algorithms are computationally expensive and become problematic when documents are accessed frequently. The above schemes lack a balance between privacy and overhead. In our model, we achieve a good balance between search functionality, privacy and efficiency without resorting to blockchain technology. In the future, we will consider applying searchable cryptography in smart car driving [22-23]. In addition, there are other fields that can also be combined with searchable encryption, such as federated learning [24-27], IoT [28], etc.

Privacy of searchable encryption. Forward and backward privacy are the most general security concepts in the field of searchable encryption, and as can be seen in [29] forward security is defined for the first time. In [29], the authors pioneered the use of oblivious RAM (ORAM) and discussed how to design SSE schemes with forward security. Their construction also handles deletion in an elegant way, but its essence is to sacrifice security for scheme performance. Certainly, there are numerous additional challenges, such as [30]; however, these will not be addressed in this discussion. In 2017, Bost et al. specifically explored backward security in [31] and categorized it into three types with different levels of security, which laid the groundwork for numerous subsequent scheme [11, 32]. Based on privacy, various searchable encryption schemes with diverse search capabilities have been researched, using the characteristics of other fields [33-35] as a background, such as [36-39].

Robust of searchable encryption. Xu et al. [7] discovered robustness against searchable encryption. Roughly speaking, robustness refers to the extent to which addition and deletion operations offset each other. Most DSSE schemes do not explicitly address robustness implementations, for example [40-41]. However, [31] inadvertently achieved robustness in their scheme, albeit with slightly higher computational and communication overhead. This limitation makes their scheme unsuitable as a searchable encryption implementation for healthcare systems. On the other hand, Xu et al. [7] developed a method that constructs comprehensive historical interrogation lists and achieves robustness by utilizing a pseudo-random function with updatable keys. This technique enables ciphertext conversion between different keys, ensuring security while efficiently navigating the history of the interrogation list. However, Xu et al. [7] did not consider how to handle repeated deletion operations or how to recover the deleted file set. Side by side, the current definition of robustness is incomplete as it lacks a description of the coupling between repeated queries. For instance, performing repeated additions should result in high coupling, which favors deletion. Conversely, repeated deletion operations should exhibit low coupling, and a second delete operation should not be executed for a deletion that has already been performed.

Therefore, designing searchable encryption schemes with robustness against medical datasets generated by generative AI is a natural idea. The scheme requires that the medical dataset has a complete and analyzable data structure. This structure is utilized to design inverted indexing results that streamline the search process and guarantee both forward and backward security. Furthermore, we aspire to implement searchable encryption within smart vehicles in the future [42].

1.2 Contributions

Our research focuses on recovering indexing information for critical data that has been mistakenly deleted. Existing studies lack discussion on restoring indices for inadvertently deleted data. To address this gap, we propose a hypothesis and design a unique index structure and implementation scheme. We define this data recovery capability as *reversibility*.

The reversibility in this scheme is realized by key update pseudo-random function. In addition, this recovery process can also be called *rollback*, a term derived from transaction rollback in database systems such as MySQL. The terms *reversibility* and *rollback* will be used interchangeably in the text: the former refers to the property of recovery, while the latter describes the specific execution of the recovery behavior. Each ciphertext block in the index structure contains the latest state of the nearest node, and the query history can be recursively obtained through trapdoors and invalidated or recovered on demand. Due to the introduction of reversibility, the existing leakage functions cannot adequately prove the security of the scheme, so we design novel leakage functions to prove that the scheme satisfies both forward and backward security.

The central focus of this paper is to develop a reversible SE scheme with the following main contributions:

(1) A rollback mechanism is introduced to compensate for the lack of robustness. Based on the key updatable pseudo-random function, we implement the revocation of

the last update query. For a delete operation, the deleted file set is restored; for an add operation, the information of the added files is invalidated. After the rollback, the connectivity between the master branch and the bridge is still ensured. This feature can provide fault tolerance for edge AI-initiated operations.

(2) A newly developed indexing data structure, referred to as the Bridge Structure, is specifically designed for rollback mechanisms. This structure enables secure search operations even in the presence of trapdoors. It consists of a primary branch and a secondary branch, which together effectively meet the requirements for both robustness and rollback functionality. Its linear configuration results in comparatively low search time complexity and minimal associated overhead, making it well-suited for deployment on lightweight devices.

(3) A scheme has been devised that adequately combines reversibility and *Bridge*. In the security proof, brand new leakage functions are introduced to prove that the scheme is backward secure. Furthermore, through sound theoretical analysis and experimental comparison, the proposed scheme is shown to be effectively deployed on lightweight devices.

2 Preliminaries

In this section, we introduce the symbolic concepts used, the system model, and the key-updatable pseudo random function.

2.1 Notations

We use $\lambda \in \mathcal{N}$ to represent the security parameter. $e \xleftarrow{\$} \mathcal{X}$ denotes the uniform selection of a pseudorandom value e from the space $\mathcal{X}$, where $|\mathcal{X}|$ is the number of elements in $\mathcal{X}$. Let $x \parallel y$ indicate the concatenation of strings x and y. Meanwhile, $\{0,1\}^{\lambda}$ represents a binary string of length λ, and $\{0,1\}^{*}$ denotes a binary string of arbitrary length. The negligible function **negl**(λ) represents the ability to output a very small probability value when inputting the parameter λ. The negligible function is often used in security proofs to indicate that two models do not have a significant difference. Details are in Table 1.

Table 1. Notations

Notations	Description
ct, *st*	Counter and search token
ind	File identifier
T	A set hold *st* information related to *ind*
P	Pseudo random permutation
F	Key-updatable pseudo random function
EDB	Encryption database
H	Hash function

2.2 System Model

The system model comprises five entities: the data owner, data user, data processing unit, documents, and a trusted cloud server, as shown in Figure 1. The data owner can collect diverse datasets, such as disease information, smart home data, and vehicle status. These datasets are provided to the data processing unit for processing, resulting in complete and usable data. The data processing unit may involve tasks such as data formatting or deep learning, which are not elaborated here as they are not the primary focus of this paper. Documents are utilized to construct the EDB and encrypted indexes. The user has three operational actions: search, update, and rollback. Search operations can only be performed upon providing a trapdoor. Update operations are similar to search operations but also involve submitting the associated encrypted document information. Rollback operations, also referred to as reversibility (with distinctions between the two explained later), require a system token to be executed.

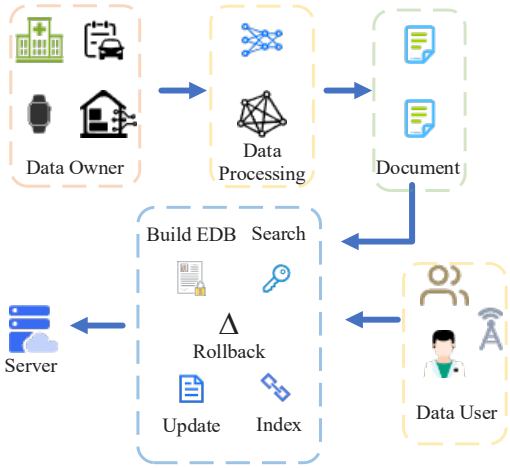

Figure 1. System model

2.3 Key-Updatable Pseudo Random Function

Pseudo random function (PRF) P is key-updatable iff there have a token that can update the PRF values from with secret key K_1 to K_2.

Definition 1 (Key-Updatable PRF). Let P: $\mathcal{Y}_P \leftarrow \mathcal{K}_P \times \mathcal{X}_P$ be an efficient PRF. P is a viable key-updatable PRF if the following property is satisfied.

There are two efficient algorithms P.UT: $\mathcal{Y}_P \leftarrow \mathcal{K}_P \times \mathcal{K}_P$ and P.UK: $\mathcal{Y}_P \leftarrow \mathcal{K}_P \times \mathcal{Y}_P$. For example, given a token $\Delta_{K_1 \to K_2}$ by P.UT(K_1, K_2), pseudo random value P(K_2, x) will equal P.UK($\Delta_{K_1 \to K_2}$, P(K_1, x)).

We use the same key-updatable PRF as described in reference [7]. A key-updatable PRF instance can be constructed as follows.

$$\begin{cases} P(K_1, x) = H(x)^{K_1} \\ \Delta_{K_1 \to K_2} = K_1^{-1} \cdot K_2 \\ P(K_2, x) = P.UK(\Delta_{K_1 \to K_2}, P(K_1, x)) \\ \qquad\quad = H(x)^{K_2} \end{cases} \tag{1}$$

3 Our Construction

In this section, we construct a dynamic searchable encryption with an executable rollback operation with backward security, and we notate the scheme as R(rollback)B(backward)-DSSE. RB-DSSE aims to achieve recovery of important data after deletion by designing a bridge structure and providing reversible properties.

3.1 Overview

This section is organized into three parts: the index structure, the reversibility property, and finally the proposed scheme.

This section is divided into three parts. In the first part, we introduce the proposed scheme which has reversibility and we design a data structure called Bridge in order to realize the reversibility. In the second part, we introduce a data structure called Bridge which consists of multiple encrypted data blocks. Each data block has two main roles. The first role is to traverse the history query record, and the second role is to compile the data blocks associated with the keyword w, as shown in Figure 2 and Figure 3. In Section III, we analyze reversibility in detail and use rollback as a concrete implementation of reversibility, which is illustrated in Figure 3.

Both reversible and rollback will appear below, both of which point to reversibility, except that reversible will appear in the description of this concept and not in the specific behavioral action, while rollback will appear in our specific behavioral description of reversibility to explain how reversibility is guaranteed. There is no need to make a distinction between the two.

3.2 Complete Scheme

In this section, we introduce RB-DSSE, Reversible DSSE with forward and backward privacy. This scheme is designed to provide the ability to rollback while ensuring forward and backward security.

Setup: The first line creates three empty mappings, *EDB*, $\sum$, and *T*. *EDB* denotes encrypted, EDB denotes encrypted database, $\sum[w]$ denotes current latest *st* and counter *ct* of the keyword *w*, and *T*[*ind*] is used to denote the most recent *st* on which the document *ind* was operated. K_w is the secret key. The second line sets the pseudo-random function and key *F* to be used throughout the algorithm and the updated pseudo-random function F_1. The details of the content related to initialization can be seen in Algorithm 1. The details of these symbols will be seen in subsequent algorithms.

Table 2. Details of setup algorithm

Algorithm 1. Setup
Input: security parameter λ
Output: Send *EDB* to Server
1: $EDB, \sum, T \leftarrow$ empty map
2: Initialize a normal PRF function F and key-updatable PRF function F_1
3: $K_w, K_s \leftarrow \{0,1\}^{\lambda}$

Update: From line 1 to line 8, it is to compute the search token for the new node. First get the last saved st_{ct} and *ct* from $\sum$. If st_{ct} is $\perp$, it indicates that *w* has not been requested for updates. Using k_{ct+1} and st_{ct} to get new search token st_{ct+1} and stored (st_{ct+1}, ct + 1) to $\sum$. Lines 9 through 11 show some of the responses to the add operation, then encrypt (*op*, *ind*) using F_1 and save them as e_{ct+1}. Lines 12 through 18 show some of the responses to the delete operation and the rollback operation operations. If *op* is *del* or K'_w is $\perp$, then first form a random K'_w, then $\Delta \leftarrow K_w^{-1} \cdot K'_w$ is constructed, and finally Δ is used with other data to form the cipher text $e_{ct+1} \leftarrow F_1(K_w, st_{ct+1}) \oplus (op||\Delta)$. Lines 19 through 21, are used to update the mapping with the results of the new calculations. Lines 22 through 28, construct links between nodes targeting the same file *ind* at different times. The latest search token *st* and *ct* related to *ind* are obtained through $T[ind_i]$. Taking Figure 2 as an example. If the insertion of node st_3 is currently being performed, then st_1 and $c_1^{ind_1}$ can be obtained through $T[ind_i]$. Finally, $T[ind_i]$ needs to be updated. For example, in Figure 3, $c_1^{ind_1}$ points to st_0. If operation of node st_3 is *add*, $T[ind_1]$ saves st_3 and $c_3^{ind_1}$. If the operation is *del*, $T[ind_1]$ saves st_0 and c_0. If the operation is *rol*, element that will store in $T[ind_1]$ needs be computed using $c_1^{ind_1}$. On lines 29 to 30, the above results are packaged and sent to the Server. Details are in Table 3.

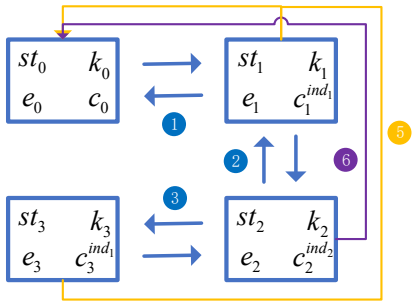

Figure 2. Updating a query

Table 3. Details of update algorithm

Algorithm 2. Update
Input: K_s, σ, *op*, (*w*, *ind*); EDB
Output: u, e_{ct+1}, k_{ct+1}, c_{ct}^{ind}
Client:
1: $(st_{ct}, ct) \leftarrow \sum[w]$
2: If $(st_{ct}, ct) = \perp$
3: $ct \leftarrow 0$, $st_0 \leftarrow \{0,1\}^{\lambda}$
4: End If
5: $t_w \leftarrow F(K_s, w)$
6: $k_{ct+1} \leftarrow \{0,1\}^{\lambda}$
7: $st_{ct+1} \leftarrow P(k_{ct+1}, st_{ct})$
8: $\sum[w] \leftarrow (st_{ct+1}, ct+1)$
9: If $op = add$
10: $e_{ct+1} \leftarrow F_1(K_w, st_{ct+1}) \oplus (add \parallel ind)$
11: $T_{add}[ind] = st_{ct+1}$
12: Else If $op = del$ or $op = rol$

13: If $op = del$ or K'_w is $\perp$
14: $K'_w \leftarrow \{0,1\}^{\lambda}$
15: End If
16: $\Delta \leftarrow K_w^{-1} \cdot K'_w$
17: $e_{ct+1} \leftarrow F_1(K_w, st_{ct+1}) \oplus (op \| \Delta)$
18: End If
19: $k_{ct+1} \leftarrow k_{ct+1} \oplus H(st_{ct+1} \| ct+1)$
20: $(st_{ct}^{ind}, c_{ct-1}^{ind}) \leftarrow T[ind]$
21: $c_{ct}^{ind} \leftarrow F_1(K_w, st_{ct+1}) \oplus (st_{ct}^{ind})$
22: If $op = del$
23: $T[ind] = (st_0, c_0)$
24: Else If $op = add$
25: $T[ind] = (st_{ct+1}, c_{ct}^{ind})$
26: Else
27: $T[ind] = (F_1(K_w, st_{ct}^{ind}) \oplus c_{ct-1}^{ind}, c_0)$
28: End If
29: $u \leftarrow H(t_w \| st_{ct+1})$
30: Send $(u, e_{ct+1}, k_{ct+1}, c_{ct}^{ind})$ to Server.
Server:
31: $EDB[u] \leftarrow (e_{ct+1}, k_{ct+1}, c_{ct}^{ind})$

Search: Lines 1 to 6, it is the client that generates t_w, st_{ct}, ct based on the keyword w and sends it to the server. In lines 9 through 11, the index u of the relevant data in the EDB database is obtained by computing t_w and st, and the data is taken out and decrypted. Lines 12 to 15, are skipped when a node is encountered that has been reversed afterwards, this is because the node has already been re-encrypted and has been invalidated, therefore the subsequent steps are skipped and go directly to the next loop. Lines 16 to 17, respectively, decrypt the encrypted message e in a valid node and determine that if op is add, then that node is added to the result. Lines 21 through 34 realize different effects depending on the operation. If op is del, then the deletion is performed between the history chain lists. If rol, then a rollback operation is performed, i.e., the reversal of the deleted operation is realized. Details are in Table 4.

There are five orders of execution that can be discussed for search queries: *add−del−search*, *add−rol−search*, *del−rol−search*, and *del−search−rol*. In the first order, if the file identifier added by *add* is *ind*, then *del* invalidates all node information associated with the *ind* (including the node of *del* itself). The only effect of the node being invalidated is to maintain the integrity of the history queries list. In the second order, when a search query occurs, node of *rol* is accessed first, and it immediately invalidates its neighboring *add* node. The third order is consistent with the second order. However, since *rol* is accessed before *del*, and *del* is invalidated by *rol* before it takes effect, the purpose of undoing the deletion operation is achieved. The search query is executed as in the first case. After rollback occurs, the state of *del* is restored by key-updatable PRF. However, at this stage, the nodes belong *add* remains invalidated and has not been recovered. When the next search query occurs, the deletion operation will be executed again because information of deletion operation has been recovered. At this time, the invalidate nodes belong *add* will be fully recovered according to the characteristics of XOR. After *rol* occurs in the process described above, $T[w]$ will be updated to the node pointed to by c^{ind} in the revoked node. For example, the node st_6 in Figure 3 represents *rol*, then the subsequent $T[ind]$ will point to st_0 according to $c_5^{ind_1}$.

Table 4. Details of search algorithm

Algorithm 3. Search
Input: K_s, $\sum$, w, EDB
Output: t_w, st_{ct}, ct, I
Client:
1: $t_w \leftarrow F(K_s, w)$
2: $(st_{ct}, ct) \leftarrow \sum[w]$
3: If $(st_{ct}, ct) = \perp$
4: Return $\varnothing$
5: End If
6: Send (t_w, st_{ct}, ct) to Server
Server:
7: $I \leftarrow \varnothing$
8: For $i = ct$ to 1
9: $u \leftarrow H(t_w \| st_i)$
10: $(e_i, k_i, c_i^{ind}) \leftarrow EDB[u]$
11: $k_i \leftarrow k_i \oplus H(st_i \| i)$
12: If $e_i \oplus F_1(K_w, st_i)$ is failed
13: $st_{i-1} \leftarrow P^{-1}(k_i, st_i)$
14: Continue
15: End If
16: $(op \| info) \leftarrow e_i \oplus F_1(t_w, st_i)$
17: If $op = add$
18: $I \leftarrow I \cup info$
19: Else If $op = del$ or $op = rol$
20: $st^{ind} \leftarrow st_i$, $times \leftarrow 0$
21: While $st^{ind} \neq st_0$
22: $f_1 \leftarrow F_1(K_w, st^{ind})$
23: $u \leftarrow H(t_w \| st^{ind})$
24: $(e_i^{ind}, k^{ind}, c_i^{ind}) \leftarrow EDB[u]$
25: $e_i^{ind} \leftarrow e_i^{ind} \oplus F_1.UK(info, f_1)$
27: $st^{ind} \leftarrow c_i^{ind} \oplus f_1$
28: $c_i^{ind} \leftarrow c_i^{ind} \oplus F_1.UK(info, f_1)$
29: $EDB[u] \leftarrow (e_i^{ind}, k_i^{ind}, c_i^{ind})$
30: times++
31: If times ≥ 2
32: $st^{ind} \leftarrow st_0$
33: End If
34: End While
35: End If
36: End For
37: Return I

3.3 Bridge Structure

The index structure is based on [32], which transforms the asymmetric encryption structure into a symmetric encryption structure. While [32] maintains a list only for historical queries operations, the data structure also includes a branch for the same document ind_i on keyword w. Figure 3 denotes the properties of bridge structure, where st_i denotes search token; F_1 denotes key-updatable PRF; e_i, c_i, k_i denote details of node on st_i; P denotes pseudo random permutation.

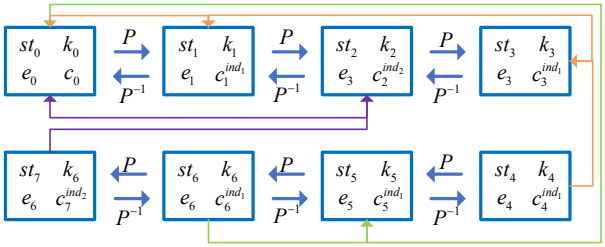

Figure 3. Bridge structure

In Figure 3, the update operations (add, ind_1), (add, ind_2), (add, ind_1), (del, ind_1), (del, ind_1), (rol) and (add, ind_2) are performed in chronological. The diagram illustrates all the properties of the bridge structure, where rol is the rollback feature of the structure. Each node consists of three parts: (st_i, k_i), e_i and c_i. (st_i, k_i) is used to maintain the two nodes that are temporally connected backward and forward in the history query list, e_i represents the document-related encrypted information, and c_i is used to connect tow nodes that contain the same ind_i. Additionally, st_0 denotes the default node to which all other nodes can be connected. The connection information between nodes can be recognized only when the client provides a token.

For insert operation, the e_i include document ind_i and update information add. If the document ind_i does not currently exist in the current history list, c_i^{ind} will point to c_0, otherwise c_i^{ind} points to the position of the last c_i^{ind} when document was last added. For delete operation, the e_i include update information del and a update token Δ. For document ind_1, if the last update was insertion, then c_1 will point to the node where the last insertion wasp performed on ind_i. If no information related to ind_i exists in the current history list, then c_i will point to c_0. The role of Δ is to provide a token that invalidates documents that haven been added between deletion operations. The search operation traverses the entire history list. The nodes encountered will exist in three states: invalidated nodes, valid nodes, and nodes that are about to have their state rolled back. The definition of rollback and the details of the various queries will be provided below.

3.4 Reversibility

In this section, the definition and implementation of reversibility are presented. And Figure 3 is used as a concrete example to incrementally illustrate the specific operation of rollback.

Definition 2 (DSSE with reversibility). Let $\sum$ be a DSSE scheme. $\sum$ is said to be a reversible searchable encryption scheme if it allows the user to undo the effects of the last update query and the update query itself, but still maintains security and correctness.

As shown in Figure 3, after repeating the addition to ind_1, a deletion operation followed. There are two issues to consider regarding how to achieve robustness. The first issue is how to handle repeated queries, such as st_1 and st_3. We do not make independent considerations because the two nodes are insertion operations on the same document, but they are added to the history query list normally. The second issue is how to remove duplicate documents using a single update operation. In each node contains attribute $c_i^{\mathrm{ind}_i}$ which serves to connect the temporally proximate nodes on the keyword w. This process can be referenced in Figure 3 with the yellow line (the green and purple colors are also indicative of this process).

Update operations such as st_4 do not execute deletions during the update phase, but actual deletion takes place in the search phase. e_4 in the delete operation differs from the insert operation in that e_4 carries a token Δ. This token invalidates the information of the specified nodes sequentially, based on point $c_4^{ind_1}$. Specifically, the secondary encryption is performed using a key-updatable PRF, but will not satisfy rollback if invalidated as a random string.

Similar to the deletion operation, the rollback event is initiated in the update query, but is processed during the search query. The rollback event is responsible for undoing the most recent update query that occurred. For insertion operation st_i, the affected node include only st_i. For deletion operation st_i, the affected nodes include all the nodes that were deleted by st_i. The key-updatable PRF is still used as the primary method in the rollback event. After a search operation is completed, rollback events and the update queries are fully processed. This also signifies that any Δ that have been generated must be invalidated to prevent them from being retrieved by the server.

4 Security Analysis

This section shows a detailed security analysis. Forward and backward security is proved firstly, and secondly, it is shown that the scheme satisfies the security of CKA-2.

4.1 Forward and Backward Privacy

Prove that the scheme satisfies forward and backward privacy by introducing some auxiliary functions.

Let Q be a list to store each query. Q consists of two entities, (t, op, (w, ind)) and (t, w), denoting updates and searches, respectively.

The auxiliary function $TimeDB(w)$ is designed to retrieve the set of document identifiers associated with the keyword w, ensuring that these identifiers have not been removed from the database. Its function is defined as follows.

$$TimeDB(w) = \{(t_i, ind) \mid (t_i, add, w, ind) \in Q \text{ and } \forall t_j : (t_j, del, (w, ind))) \notin Q\} \quad (2)$$

According to the above definition, for a searchable encryption scheme with reversible properties, its $TimeDB(w)$ can be defined as follows.

$$TimeDB(w) = \{(t_i, ind) \mid (t, add, (w, ind)) \in Q \text{ and } \forall t_j : (t_j, del, (w, ind)) \notin Q \text{ and } \forall t_z : (t_z, rol, w) \notin Q\} \quad (3)$$

The auxiliary function $Update(w)$ is used to represent the set of timestamps that occur each time an update operation occurs on a document. Its function is defined as follows.

$$Update(w) = \{t_i \mid \exists (op, ind) : (t_i, op, (w, ind)) \in Q\} \quad (4)$$

The update operations described above contain only additions and deletions. Whereas, in our scheme, reversible operations are added to update operations.

$$Update(w) = \{t_i \mid \exists (op, ind) : (t_i, op, (w, ind)) \in Q \text{ or } \exists (rol, ind) : (t_j, rol, (w, ind)) \in Q\} \quad (5)$$

Based on the above auxiliary functions, forward and backward security can be formally defined. The definition starts with forward security.

Definition 3 (Forward Privacy). If the proposed scheme is forward privacy, then its leakage function can be defined as:

$$\mathcal{L}^{Updt}(op, w, ind) = \mathcal{L}'(op, \{t_i, ind\}) \quad (6)$$

where $\mathcal{L}'$ is a stateless function.

In the scheme, the new query state, which can be obtained through $st_{ct+1} \leftarrow P(k_{ct+1}, st_{ct})$, and going through the new state to get the previous ciphertext needs to go through $st_{i-1} \leftarrow P^{-1}(k_i, st_i)$, which means, the adversary needs to crack P^{-1}. Hence the forward security of the scheme is ensured through the pseudo-random permutation.

Definition 4 (Backward Privacy). If the proposed scheme is backward privacy, then its leakage function can be defined as:

$$\begin{cases} \mathcal{L}^{Updt}(op, w, ind) = \mathcal{L}'(op, w) \\ \mathcal{L}^{Srch}(w) = \mathcal{L}''(TimeDB(w), Updates(w)) \end{cases} \quad (7)$$

where $\mathcal{L}'$ and $\mathcal{L}''$ are stateless functions.

This is a standard backward privacy definition. According to the definition, in a nutshell, backwards privacy ensures that added and deleted indexes do not reveal any information when searched. In the proposed scheme, the index is encrypted by $k_{ct+1} \oplus H(st_{ct+1}||ct+1)$, which is a ciphertext, and an adversary cannot obtain the corresponding index without a pointer *ct* saved by the user. Thus it is satisfying backward security.

4.2 CKA-2 Security Analysis

Existing SSE all leak some information to the server in order to strike a balance between efficiency and security. Therefore, the security of searchable encryption is defined as not leaking more information than the specified leakage limit. Security is implemented by providing the simulator with a set of leakage functions $\mathcal{L} = \mathcal{L}^{Setup}, \mathcal{L}^{Srch}, \mathcal{L}^{Updt})$.

Definition 5 (Adaptively Secure Searchable Encryption). Let $\sum$ be a searchable encryption scheme that comprises three algorithms: (Setup, Search, Update). There exists an adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ capable of utilizing the leakage function $\mathcal{L} = \mathcal{L}^{Setup}, \mathcal{L}^{Srch}, \mathcal{L}^{Updt})$. There are two probabilistic experiments as follows.

(1) $\mathbf{Real}_{\mathcal{A}}^{\sum}(\lambda)$: Let $\mathcal{A}$ chooses a database DB, the implementation executes Setup(λ, DB, $\perp$) and return EDB to $\mathcal{A}$. Then, $\mathcal{A}$ will adversary adaptively choose search queries or update queries q_i. For search query q_i, the implementation will answer q_i by running $((\sigma, \mathrm{DB}_i(w_i)), \mathrm{EDB}_{i+1}) \leftarrow \mathrm{Search}(K, \sigma_i, q_i; \mathrm{EDB}_i)$. For update query q_i, the implementation will answer the query by $(\sigma_{i+1}, \mathrm{EDB}_{i+1}) \leftarrow \mathrm{Update}(K, \sigma_i, q_i; \mathrm{EDB}_i)$. Finally, $\mathcal{A}$ will outputs a bit $b \in \{0,1\}$.

(2) $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\sum}(\lambda)$: Let $\mathcal{A}$ chooses a database DB. The simulator $\mathcal{S}$ uses $\mathcal{L}^{Setup}$ to get EDB $\leftarrow \mathcal{S}(\mathcal{L}^{Setup}(DB))$ and return it to adversary $\mathcal{A}$. Then, $\mathcal{A}$ will adversary adaptively choose search queries or update queries q_i. For search query q_i, the simulator will answer the q_i by running $\mathcal{S}(\mathcal{L}^{Srch}(q_i))$. For update query q_i, the simulator will answer the query by running $\mathcal{S}(\mathcal{L}^{Updt}(q_i))$. Finally, the adversary $\mathcal{A}$ outputs a bit $b \in \{0,1\}$.

For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, if there exists a PPT simulator $\mathcal{S}$ that meets the following conditions.

$$|\Pr(\mathbf{Real}_{\mathcal{A}}^{\sum}(\lambda) = 1) - \Pr(\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}^{\sum}(\lambda))| \leq \mathbf{negl}(\lambda) \quad (8)$$

Theorem 1. Let F be PRF, F_1 be key-updatable PRF, and P be a PRR. Define leakage $\mathcal{L} = \mathcal{L}_{Setup}, \mathcal{L}_{Srch}, \mathcal{L}_{Updt})$ as follows.

$$\begin{cases} \mathcal{L}_{Setup} = \perp \\ \mathcal{L}^{Srch}(w) = (\mathrm{TimeDB}(w), \mathrm{DelHist(w)}) \\ \mathcal{L}^{Updt}(\mathrm{op}, w, \mathrm{ind}) = (\mathrm{op}, \mathrm{ind}) \end{cases} \quad (9)$$

Then RB-DSSE is a L−adaptively-secure DSSE with forward and backward privacy.

Proof. The proof consists of a series of games from $\mathbf{Real}^{\sum}_{\mathcal{A}}(\lambda)$ to $\mathbf{Ideal}^{\sum}_{\mathcal{A},\mathcal{S}}(\lambda)$. By leveraging the transferability of indistinguishability, we can establish the indistinguishable boundary between $\mathbf{Real}^{\sum}_{\mathcal{A}}(\lambda)$ and $\mathbf{Ideal}^{\sum}_{\mathcal{A},\mathcal{S}}(\lambda)$.

Game G_0: The game is exactly same as $\mathbf{Real}^{\sum}_{\mathcal{A}}(\lambda)$. Then we have

Game G_1: The game is the same as G_0 except that T_w instead of F to generate t_w. T_w is a map for mapping $h(w)$ to t_w. Thus t_w is no longer obtained through F, but directly T_w. When t_w is needed, T_w is checked, if the corresponding $h(x)$ does not exist in TW, then randomize generated $h(x)$ and save it to T_w, otherwise return directly. Thus, if adversary $\mathcal{A}$ can distinguish between G_1 and G_0, we can create an adversary $\mathcal{B}_1$ to distinguish between pseudo-random functions and truly random functions. Thus we have

$$|\Pr[G_0=1]-\Pr[G_1=1]|\le \mathbf{Adv}^{PRF}_{F,\mathcal{B}_1}(\lambda). \tag{10}$$

Game G_2: The game is the same as G_1 except that the obtaining of k_i. The experiment also requires the maintenance of a map KEY. If k_i was generated, KEY need to store it by KEY[$(w \parallel i)$] $\rightarrow k_i$. Since K_i is supposed to be a random string and is now preserved through KEY, G_2 and G_1 are distinguished with the same probability. Then we have

$$\Pr[G_2=1]=\Pr[G_1=1]. \tag{11}$$

Game G_3: The game is same as G_2 except instead of H in the update protocol. Replacing H with a random strings. In search query, random string will replace the output of the oracle. Specifically, replace $u \leftarrow H(t_w \parallel st_{c+1})$ with $u \leftarrow \{0,1\}^{\lambda}$. Also, there will still be a need to use the already generated u in subsequent operations, so a table T_u is needed to store the u. Thus, if adversary $\mathcal{A}$ can distinguish between G_3 and G_1, we can create an adversary $\mathcal{B}_2$ to distinguish random oracle and truly random functions. Thus we have

$$|\Pr[G_3=1]-\Pr[G_2=1]|\le \mathbf{Adv}^{H}_{F,\mathcal{B}_2}(\lambda). \tag{12}$$

Simulator: Now, we can use leakage functions to simulate the queries. However, that the simulator can only use data that is allowed through by the leakage functions. The simulator can use *TimeDB*(w) and *DelHist*(w) in $\mathcal{L}^{Srch}$ as inputs to search protocol, followed by game G_3 being constructed by the simulator $\mathcal{S}$. Then there are

$$\Pr[G_4=1]=\mathbf{Ideal}^{\sum}_{\mathcal{A},\mathcal{S}}(\lambda). \tag{13}$$

Finally, we can conclude that the advantage of adversary $\mathcal{A}$ for attacking RB-DSSE is

$$\begin{cases} \mathbf{Adv}^{PRF}_{F,B_1}(\lambda)+\mathbf{Adv}^{PRF}_{F,B_2}(\lambda)=\mathbf{Adv}^{PRF}_{F,B_1+B_2}(\lambda) \\ \mathbf{Adv}^{PRF}_{F,B_1+B_2}(\lambda)\le \mathbf{negl}(\lambda) \\ |\,\mathbf{Real}^{\sum}_{\mathcal{A}}(\lambda)-\mathbf{Ideal}^{\sum}_{\mathcal{A},\mathcal{S}}(\lambda)\,|\le \mathbf{Adv}^{PRF}_{F,\mathcal{B}_1+\mathcal{B}_2}(\lambda). \end{cases} \tag{14}$$

5 Implementation and Evaluation

The experiment using C++ was deployed on Ubuntu 22.04.2 with an Intel Core i5-12490F, 3.00GHz processor, and 16GB RAM, utilizing the Enron email dataset, OpenSSL, and Boost. We randomly selected 3190 documents from the Enron dataset and extracted 34546 different keywords from the documents using specified separators.

In our experimental comparison, RB-DSSE is evaluated against three other schemes: ROSE [7], FAST [9], and SR-DSSE [43]. The selection of comparison programs is based on two main principles: similarity of index structure and functionality. For example, FAST, RB-DSSE have similar index structures. And ROSE, SR-DSSE are robust and realize robustness in completely different ways. ROSE stands out for its focus on achieving robustness, particularly in handling repeated update operations. In contrast, RB-DSSE focuses on error recovery. ROSE utilizes a novel cryptographic primitive, namely key-updatable pseudo-random functions, and employs a logical linked-list structure to connect different queries for the same keyword. FAST shares a similar structural architecture with RB-DSSE, and it prioritizes IO efficiency. SR-DSSE is the most recent scheme with a robust design. Table 2 compares the features and differences of these schemes in detail, where M is total number of keyword/file pairs and N is total number files. Details are in Table 5.

Table 5. Scheme comparison

Scheme	Type	Robust	Rollback	Storage
ROSE	Single	√	×	$\mathcal{O}(M\log N)$
FAST	Single	×	×	$\mathcal{O}(M+N)$
SR-DSSE	Single	√	×	$\mathcal{O}(M+N)$
Ours	Boolean	√	√	$\mathcal{O}(M+N)$

Figure 4 illustrates the time needed for n update queries to occur. Among the algorithms, FAST requires the least amount of time, followed by RB-DSSE. FAST, which utilizes a bitmap. RB-DSSE, on the other hand, has more key-updatable PRF compared to FAST, making it inferior in terms of update time. The remaining algorithms incorporate multiple hash algorithms and vary in the frequency of PRF operations, resulting in longer processing times.

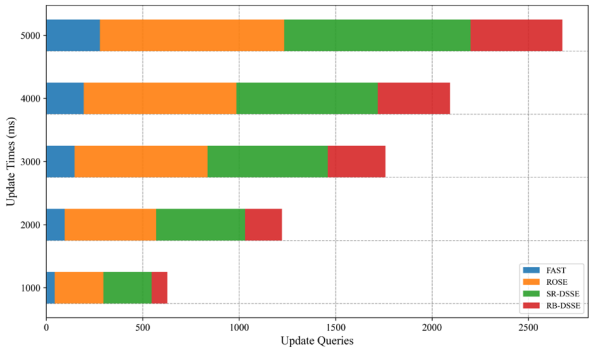

Figure 4. Time consumption for updates

Figure 5 illustrates the search time across various dataset sizes for a constant number of searches. FAST is the most efficient as it saves the current search result after each search and utilizes the existing set for subsequent searches, significantly reducing time spent on repetitive operations. While RB-DSSE mainly traverses the main branch and only navigates the bridge when encountering *rol* and *del*, where *del* is specifically for duplicate files and *rol* only involves two nodes, the time taken by RB-DSSE is primarily determined by the traversal of the main branch. While ROSE will analyze whether it is updatable in each node and will frequently utilize the pseudo-random function for key updatability, this process takes longer. The search query in SR-DSSE involves two rounds of communication, with the main content executed on the client side. This execution is limited by hardware, thus affecting the overall execution time.

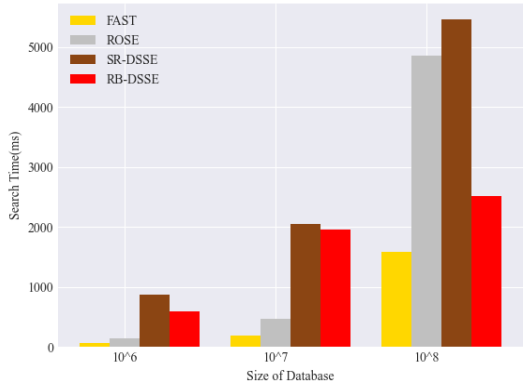

Figure 5. Time cost for obtaining different result sets

Figure 6 illustrates the time needed to execute a search query following multiple delete operations. It can be seen that FAST is the fastest, followed by RB-DSSE. FAST performs only one traversal of the data structure to complete the job. During the search phase, the actual deletion of RB-DSSE takes place, and the update occurs when the delete operation merges the node in the master branch and bridge. Duplicate deletions are ignored during the search, and the first actual deletion is executed. ROSE conducts decryption and key updating each time a deletion is encountered, which contributes to the extended time required by the scheme. While SR-DSSE requires two rounds of communication and places the main workload on the user side, the communication time and hardware setup are limiting factors for its performance.

According to the experimental comparison graph, it can be seen that the efficiency of RB-DSSE is not the highest, this is because the FAST scheme, which has the highest efficiency, has researched a caching mechanism, if between the two search operations, no update operation has occurred, then the results are returned directly; otherwise, only the updates between the two searches are searched for and merged into the existing set of results.

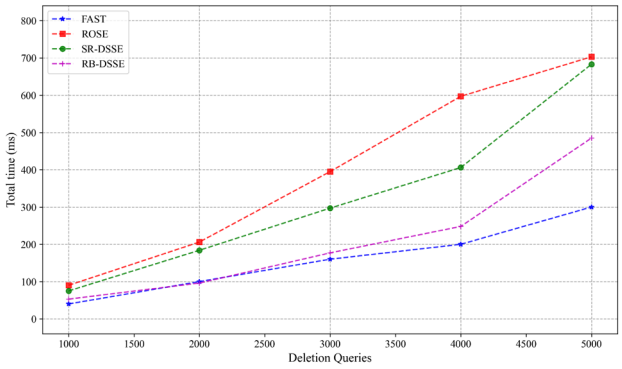

Figure 6. Maintain robustness time consumption

6 Conclusion

In summary, our proposed RB-DSSE scheme addresses the challenge of unrecoverable sensitive data after accidental deletion, not only electronic medical records but also any dataset suitable for indexing, by taking medical data as an entry point. By introducing a novel reversibility feature, we enhance the robustness of the system, enabling efficient recovery from erroneous operations while maintaining connectivity between the master branch and the bridging structure. The innovative bridging structure we developed ensures the safety of search operations, even in cases involving trapdoors.

Our comprehensive security analysis confirms that the RB-DSSE scheme meets both forward and backward privacy requirements, thus ensuring the confidentiality and integrity of various data types. Theoretical and experimental evaluations demonstrate the practical applicability of the scheme in real-world environments, making it an important solution for enhancing data security in a wide range of applications. Future work will focus on further optimizing the scheme and exploring its integration with emerging technologies to continuously address evolving security challenges in various domains.

Acknowledgments

The work is supported by the National Key R&D Program of China (No. 2023YFB2703700), these National Natural Science Foundation of China (Nos. U21A20465, 62402446, 62402444, 62172292, 62441228), the Zhejiang Provincial Natural Science Foundation of China (Nos.

LQ24F020011, LQ24F020012), the Program for Leading Innovative Research Team of Zhejiang Province (No. 2023R01001), the Fundamental Research Funds of Zhejiang Sci-Tech University under Grants (No. 22222266-Y), the "Pioneer" and "Leading Goose" R&D Program of Zhejiang (No. 2023C01119), and the Open Research Fund of Guangdong Key Laboratory of Blockchain Security, Guangzhou University.

References

[1] X. Zhang, W. Wang, P. Xu, L. T. Yang, K. Liang, High Recovery with Fewer Injections: Practical Binary Volumetric Injection Attacks Against Dynamic Searchable Encryption, *The 32nd USENIX Security Symposium*, Anaheim, CA, USA, 2023, pp. 5953-5970.

[2] Y. Wang, J. Wang, X. Chen, Secure Searchable Encryption: A Survey, *Journal of Communications and Information Networks*, Vol. 1, No. 4, pp. 52-65, December, 2016. https://doi.org/10.1007/BF03391580

[3] Z. Gui, K. G. Paterson, S. Patranabis, Rethinking Searchable Symmetric Encryption, *The 2023 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, 2023, pp. 1401-1418. https://doi.org/10.1109/SP46215.2023.10179460

[4] S. Yin, H. Li, L. Teng, A. A. Laghari, V. V. Estrela, Attribute-Based Multiparty Searchable Encryption Model for Privacy Protection of Text Data, *Multimedia Tools and Applications*, Vol. 83, pp. 45881-45902, May, 2024. https://doi.org/10.1007/s11042-023-16818-4

[5] K. Zhang, J. Long, X. Wang, H. N. Dai, K. Liang, M. Imran, Lightweight Searchable Encryption Protocol for Industrial Internet of Things, *IEEE Transactions on Industrial Informatics*, Vol. 17, No. 6, pp. 4248-4259, June, 2021. https://doi.org/10.1109/TII.2020.3014168

[6] Y. Guo, C. Zhang, C. Wang, X. Jia, Towards Public Verifiable and Forward-Privacy Encrypted Search by Using Blockchain, *IEEE Transactions on Dependable and Secure Computing*, Vol. 20, No. 3, pp. 2111-2126, May-June, 2023. https://doi.org/10.1109/TDSC.2022.3173291

[7] P. Xu, W. Susilo, W. Wang, T. Chen, Q. Wu, K. Liang, ROSE: Robust Searchable Encryption with Forward and Backward Security, *IEEE Transactions on Information Forensics and Security*, Vol. 17, pp. 1115-1130, March, 2022. https://doi.org/10.1109/TIFS.2022.3155977

[8] T. Moataz, A. Shikfa, Boolean Symmetric Searchable Encryption, *The 8th ACM Special Interest Group on Security, Audit and Control Symposium on Information, Computer and Communications Security*, Hangzhou, China, 2013, pp. 265-276. https://doi.org/10.1145/2484313.2484347

[9] X. Song, C. Dong, D. Yuan, Q. Xu, M. Zhao, Forward Private Searchable Symmetric Encryption with Optimized I/O Efficiency, *IEEE Transactions on Dependable and Secure Computing*, Vol. 17, No. 5, pp. 912-927, September-October, 2020. https://doi.org/10.1109/TDSC.2018.2822294

[10] K. He, J. Chen, Q. Zhou, R. Du, Y. Xiang, Secure Dynamic Searchable Symmetric Encryption with Constant Client Storage Cost, *IEEE Transactions on Information Forensics and Security*, Vol. 16, pp. 1538-1549, October, 2020. https://doi.org/10.1109/TIFS.2020.3033412

[11] B. Chen, T. Xiang, D. He, H. Li, K. K. R. Choo, BPVSE: Publicly Verifiable Searchable Encryption for Cloud-Assisted Electronic Health Records, *IEEE Transactions on Information Forensics and Security*, Vol. 18, pp. 3171-3184, May, 2023. https://doi.org/10.1109/TIFS.2023.3275750

[12] L. Chen, W. K. Lee, C. C. Chang, K. K. R. Choo, N. Zhang, Blockchain Based Searchable Encryption for Electronic Health Record Sharing, *Future Generation Computer Systems*, Vol. 95, pp. 420-429, June, 2019. https://doi.org/10.1016/j.future.2019.01.018

[13] R. Zhang, R. Xue, L. Liu, Searchable Encryption for Healthcare Clouds: A Survey, *IEEE Transactions on Services Computing*, Vol. 11, No. 6, pp. 978-996, November-December, 2018. https://doi.org/10.1109/TSC.2017.2762296

[14] R. Walid, K. P. Joshi, S. G. Choi, D. Y. Kim, Cloud-Based Encrypted EHR System with Semantically Rich Access Control and Searchable Encryption, The *2020 IEEE International Conference on Big Data*, Atlanta, GA, USA, 2020, pp. 4075-4082. https://doi.org/10.1109/BigData50022.2020.9378002

[15] S. Kamara, C. Papamanthou, T. Roeder, Dynamic Searchable Symmetric Encryption, *The 2012 ACM Conference on Computer and Communications Security*, Raleigh, North Carolina, USA, 2012, pp. 965-976. https://doi.org/10.1145/2382196.2382298

[16] X. Yang, T. Li, R. Liu, M. Wang, Blockchain-Based Secure and Searchable EHR Sharing Scheme, *The 4th International Conference on Mechanical, Control and Computer Engineering*, Hohhot, China, 2019, pp. 822-825. https://doi.org/10.1109/ICMCCE48743.2019.00188

[17] W. Li, L. Xu, Y. Wen, F. Zhang, Conjunctive Multi-Key Searchable Encryption with Attribute-Based Access Control for EHR Systems, *Computer Standards & Interfaces*, Vol. 82, Article No. 103606, August, 2022. https://doi.org/10.1016/j.csi.2021.103606

[18] L. Xu, Z. Sun, W. Li, H. Yan, Delegatable Searchable Encryption with Specified Keywords for EHR Systems, *Wireless Networks*, Vol. 30, No. 5, pp. 3547-3559, July, 2024. https://doi.org/10.1007/s11276-020-02410-3

[19] C. Li, M. Dong, J. Li, G. Xu, X. B. Chen, W. Liu, K. Ota, Efficient Medical Big Data Management with Keyword-Searchable Encryption in Healthchain, *IEEE Systems Journal*, Vol. 16, No. 4, pp. 5521-5532, December, 2022. https://doi.org/10.1109/JSYST.2022.3173538

[20] Y. Liu, X. Hao, W. Ren, R. Xiong, T. Zhu, K. K. R. Choo, G. Min, A Blockchain-Based Decentralized, Fair and Authenticated Information Sharing Scheme in Zero Trust Internet-of-Things, *IEEE Transactions on Computers*, Vol. 72, No. 2, pp. 501-512, February, 2023. https://doi.org/10.1109/TC.2022.3157996

[21] S. Niu, M. Song, L. Fang, F. Yu, S. Han, C. Wang, Keyword Search over Encrypted Cloud Data Based on Blockchain in Smart Medical Applications, *Computer Communications*, Vol. 192, pp. 33-47, August, 2022. https://doi.org/10.1016/j.comcom.2022.05.018

[22] H. Tan, W. Zheng, P. Vijayakumar, K. Sakurai, N. Kumar, An Efficient Vehicle-Assisted Aggregate Authentication Scheme for Infrastructure-Less Vehicular Networks, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 24, No. 12, pp. 15590-15600, December, 2023. https://doi.org/10.1109/TITS.2022.3176406

[23] H. Tan, W. Zheng, Y. Guan, R. Lu, A Privacy-Preserving Attribute-Based Authenticated Key Management Scheme for Accountable Vehicular Communications, *IEEE Transactions on Vehicular Technology*, Vol. 72, No. 3, pp. 3622-3635, March, 2023.
https://doi.org/10.1109/TVT.2022.3220410

[24] L. Zhang, T. Zhu, H. Zhang, P. Xiong, W. Zhou, FedRecovery: Differentially Private Machine Unlearning for Federated Learning Frameworks, *IEEE Transactions on Information Forensics and Security*, Vol. 18, pp. 4732-4746, July, 2023.
https://doi.org/10.1109/TIFS.2023.3297905

[25] H. Chen, T. Zhu, T. Zhang, W. Zhou, P. S. Yu, Privacy and Fairness in Federated Learning: on The Perspective of Tradeoff, *ACM Computing Surveys*, Vol. 56, No. 2, pp. 1-37, February, 2024.
https://doi.org/10.1145/3606017

[26] G. Zhang, B. Liu, T. Zhu, M. Ding, W. Zhou, PPFed: A Privacy-Preserving and Personalized Federated Learning Framework, *IEEE Internet of Things Journal*, Vol. 11, No. 11, pp. 19380-19393, June, 2024.
https://doi.org/10.1109/JIOT.2024.3360153

[27] X. Wang, J. Li, Z. Liu, Q. Tang, X. Wang, Enabling Secure Cross-Modal Search over Encrypted Data via Federated Learning, *IEEE Internet of Things Journal*, Vol. 12, No. 2, pp. 1933-1945, January, 2025.
https://doi.org/10.1109/JIOT.2024.3464760

[28] Q. Ma, H. Tan, T. Zhou, Mutual Authentication Scheme for Smart Devices in IoT-Enabled Smart Home Systems, *Computer Standards & Interfaces*, Vol. 86, Article No. 103743, August, 2023.
https://doi.org/10.1016/j.csi.2023.103743

[29] R. Bost, ∑οφος: Forward Secure Searchable Encryption, *The 2016 ACM Special Interest Group on Security, Audit and Control Conference on Computer and Communications Security*, Vienna, Austria, 2016, pp. 1143-1154.
https://doi.org/10.1145/2976749.2978303

[30] F. Hu, J. Shen, P. Vijayakumar, Side-Channel Attacks Based on Multi-Loss Regularized Denoising AutoEncoder, *IEEE Transactions on Information Forensics and Security*, Vol. 19, pp. 2051-2065, December, 2023.
https://doi.org/10.1109/TIFS.2023.3343947

[31] R. Bost, B. Minaud, O. Ohrimenko, Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives, *The 2017 ACM Special Interest Group on Security, Audit and Control Conference on Computer and Communications Security*, Dallas, Texas, USA, 2017, pp. 1465-1482.
https://doi.org/10.1145/3133956.3133980

[32] S. F. Sun, R. Steinfeld, S. Lai, X. Yuan, A. Sakzad, J. Liu, S. Nepal, D. Gu, Practical Non-Interactive Searchable Encryption with Forward and Backward Privacy, *Usenix Network and Distributed System Security Symposium 2021*, Virtual Event, USA, 2021, pp. 1-18.
https://doi.org/10.14722/ndss.2021.24162

[33] Y. Li, J. Shen, S. Ji, Y. H. Lai, Blockchain-Based Data Integrity Verification Scheme in AIoT Cloud–Edge Computing Environment, *IEEE Transactions on Engineering Management*, Vol. 71, pp. 12556-12565, April, 2023.
https://doi.org/10.1109/TEM.2023.3262678

[34] M. Yuan, H. Tan, W. Zheng, P. Vijayakumar, F. Alqahtani, A. Tolba, A Robust ECC-Based Authentication and Key Agreement Protocol for 6G-Based Smart Home Environments, *IEEE Internet of Things Journal*, Vol. 11, No. 18, pp. 29615-29627, September, 2024.
https://doi.org/10.1109/JIOT.2024.3392498

[35] Y. Wang, S. F. Sun, J. Wang, J. K. Liu, X. Chen, Achieving Searchable Encryption Scheme with Search Pattern Hidden, *IEEE Transactions on Services Computing*, Vol. 15, No. 2, pp. 1012-1025, March-April, 2022.
https://doi.org/10.1109/TSC.2020.2973139

[36] L. Chen, J. Li, J. Li, Toward Forward and Backward Private Dynamic Searchable Symmetric Encryption Supporting Data Deduplication and Conjunctive Queries, *IEEE Internet of Things Journal*, Vol. 10, No. 19, pp. 17408-17423, October, 2023.
https://doi.org/10.1109/JIOT.2023.3274390

[37] H. Li, T. Wang, Z. Qiao, B. Yang, Y. Gong, J. Wang, G. Qiu, Blockchain-Based Searchable Encryption with Efficient Result Verification and Fair Payment, *Journal of Information Security and Applications*, Vol. 58, Article No. 102791, May, 2021.
https://doi.org/10.1016/j.jisa.2021.102791

[38] P. Chaudhari, M. L. Das, Keysea: Keyword-Based Search with Receiver Anonymity in Attribute-Based Searchable Encryption, *IEEE Transactions on Services Computing*, Vol. 15, No. 2, pp. 1036-1044, March-April, 2022.
https://doi.org/10.1109/TSC.2020.2973570

[39] F. Meng, L. Cheng, STR-ABKS: Server-Aided Traceable and Revocable Attribute-Based Encryption with Keyword Search, *IEEE Internet of Things Journal*, Vol. 11, No. 7, pp. 12649-12659, April, 2024.
https://doi.org/10.1109/JIOT.2023.3335624

[40] Z. Zhang, Y. Wang, R. Li, H. Wang, J. Zhang, Fault-Tolerant Verifiable Encrypted Image Search With Dynamic Updates in Multi-User Cloud Environments, *IEEE Transactions on Dependable and Secure Computing*, Vol. 23, No. 4, pp. 8276-8291, July-August, 2026.
https://doi.org/10.1109/TDSC.2026.3681955

[41] S. Lv, H. Tan, W. Zheng, T. Zhang, M. Wang, A Dynamic Conjunctive Keywords Searchable Symmetric Encryption Scheme for Multiple Users in Cloud Computing, *Computer Communications*, Vol. 209, pp. 239-248, September, 2023.
https://doi.org/10.1016/j.comcom.2023.07.008

[42] H. Tan, W. Zheng, P. Vijayakumar, Secure and Efficient Authenticated Key Management Scheme for UAV-Assisted Infrastructure-Less IoVs, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 24, No. 6, pp. 6389-6400, June, 2023.
https://doi.org/10.1109/TITS.2023.3252082

[43] H. Dou, Z. Dan, P. Xu, W. Wang, S. Xu, T. Chen, H. Jin, Dynamic Searchable Symmetric Encryption With Strong Security and Robustness, *IEEE Transactions on Information Forensics and Security*, Vol. 19, pp. 2370-2384, January, 2024.
https://doi.org/10.1109/TIFS.2024.3350330

Biographies



Menglei Wang is currently pursuing the M.S. degree with the Nanjing University of Information Science and Technology of Computer Science. His research interests include computer and network security, cryptography and searchable encryption.



Haowen Tan is currently working with the School of Information Science and Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China. His research interests include network security, ubiquitous sensor networks, blockchain, and vehicular ad-hoc networks.



Wenying Zheng is currently working with the School of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China. Her research interests include cloud storage security, security systems, and network security.



Haoyu Ma is currently working with the School of Cyberspace Science and Technology, Beijing Jiaotong University, Beijing 100044, China, and also with the Beijing Key Laboratory of Security and Privacy in Intelligent Transportation. Beijing Jiaotong University, Beijing 100045, China. His research interests include operation system security related to mobile computing.