

CARE-IoMT: a pairing-free Certificateless Auditable Range-query support searchable Encryption scheme for IoMT

Arian Arabnouri

Tarbiat Modares University

Alireza Shafieinejad

shafieinejad@modares.ac.ir

Tarbiat Modares University

Article

Keywords:

Posted Date: October 23rd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10962897/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

CARE-IoMT: a pairing-free Certificateless Auditable Range-query support searchable Encryption scheme for IoMT

Arian Arabnouri¹ and Alireza Shafeinejad^{1*}

^{1*}Department of Electrical and Computer Engineering, Tarbiat Modares University, Jalal Al Ahmad, Tehran, 14117-13116, Tehran, Iran.

*Corresponding author(s). E-mail(s): shafeinejad@modares.ac.ir;
Contributing authors: arian.arabnouri@modares.ac.ir;

Abstract

The Internet of Medical Things (IoMT) and e-healthcare play a vital role in streamlining medical services and managing health information. However, they face major challenges in the management, secure retrieval, and protection of sensitive medical data, which are crucial to patient privacy. Searchable encryption has emerged as a promising cryptographic approach that supports big-data requirements, particularly efficient search, while preserving data security and system efficiency. Despite its advantages, its practical deployment remains hindered by several challenges, including insider keyword guessing attacks (IKGA), identity forgery, data availability, data integrity, limited encryption efficiency in IoT environments, slow search over large-scale datasets, and the need for advanced search capabilities. In this paper, we propose a scalable framework that combines a data-digest roll-up mechanism with blockchain technology to ensure public auditability, alleviate storage bottlenecks, and strengthen trust. In addition, we present a lightweight, pairing-free, certificateless asymmetric searchable encryption scheme to address computational inefficiency in IoMT and to support advanced functionalities such as efficient range queries and conjunctive search. Evaluations of computational complexity and smart-contract gas consumption demonstrate its efficiency and practical superiority.

Keywords: Certificate-less Authenticated Searchable Encryption , Blockchain-enabled data auditing , Smart Contract , range and conjunctive query , Pairing-free cryptography , Internet of Medical Things

1 Introduction

The digitalization of healthcare, driven by the increasing utilization of Electronic Health Records (EHRs) and the Internet of Medical Things (IoMT) in big data scenarios, has raised significant privacy concerns. The protection of patients' sensitive medical data against unauthorized access and breaches of confidentiality is crucial. So implementing extensive measures such as encryption and access control is essential. While traditional encryption ensures confidentiality, it faces with challenges, particularly in efficiently searching through outsourced encrypted data. Searchable encryption introduced to provide search capability. In our e-health scenario, we have two classes of users. The Patient (Data Owner) who wants to send encrypted medical data, and the Physician (Data User) who wants to perform searches, retrieve encrypted data, and decrypt them. Secure search in data is provided using an index (an encrypted keyword). The Physician can search in indices securely using a trapdoor. The Physician's private key is required for trapdoor generation. Therefore, others or forged identities cannot generate correct trapdoors and perform searches in data.

Nevertheless, searchable encryption in IoMT environments faces critical challenges such as computational inefficiency, lack of advanced search functionalities, malicious server actions, and data integrity violations. A primary limitation is that existing cryptographic constructions often rely heavily on computationally expensive bilinear pairings, which are highly inefficient for resource-constrained IoMT devices. Furthermore, trust to the server, which is a central storage component, plays an important role in a reliable system. What should be done if the Cloud Service Provider (CSP) is malicious, or if verifying data integrity is too costly? Recently, with the considerable increase in blockchain and smart contract technologies popularity, researchers utilized desirable features of blockchain to design more secure schemes. However, executing search operations directly via smart contracts introduces severe scalability bottlenecks and exorbitant gas costs. Also, existing schemes often lack support for advanced capabilities such as range queries, which are essential for practical medical data retrieval (e.g., specific blood glucose levels).

To overcome these computational, architectural, and functional shortcomings, this paper introduces a novel, pairing-free, and scalable framework. By eliminating expensive pairing operations, we achieve a lightweight certificateless asymmetric searchable encryption suitable for IoMT devices. Blockchain brings trust and fraud is getting as minimal as possible by using a data digest roll-up mechanism. Instead of executing searches directly on-chain, our framework delegates the search execution to the CSP. The smart contract acts merely as an immutable verifier for public auditing. Also, due to the advanced cryptographic design, the system natively supports complex queries such as range and conjunctive searches.

1.1 Related work

1.1.1 Classical Schemes

Boneh et al. [1] introduced the first Public Key Encryption with Keyword Search (PEKS) scheme. A primary drawback of their design is the requirement of a secure channel for the transmission of trapdoors. Deeming this requirement impractical, Baek

et al. [2] proposed a PEKS scheme that obviates the need for a secure channel. However, Yau et al. [3] subsequently demonstrated that this scheme was vulnerable to offline keyword guessing attacks. In response, Hu et al. [4] presented a PEKS model incorporating a designated server, which restricts the search capability to servers possessing the private key. Yau et al. [5] later introduced the online keyword guessing attack, wherein an attacker, by uploading a document and a corresponding index for each potential keyword, can infer the searched term by observing the returned document. To mitigate this threat, Chen [6] proposed an enhanced version of the designated server model. In this model, the server encrypts the returned documents, thereby restricting access exclusively to the Data User (DU). Nevertheless, this scheme remained vulnerable to Internal Keyword Guessing Attacks (IKGA). To address this issue, Shiraly et al. [7] developed a pairing-free authenticated PEKS scheme with a designated tester and formally proved its security against both internal and external keyword guessing attacks in the random oracle model.

1.1.2 Certificateless Cryptographic Schemes

Peng et al. [8] pioneered the first Certificateless PEKS (CL-PEKS) scheme by combining Certificate-Less Public Key Encryption (CL-PKE) with PEKS, thereby addressing the challenges of certificate management and key escrow. For dynamic multi-user environments, Chenam and Ali [9] introduced a model supporting conjunctive keyword search. Their authenticated scheme is resilient against various types of keyword guessing attacks. An improved, more efficient version of this scheme was later presented by the same authors [10], which enhances performance by eliminating bilinear pairings. Arabnouri and Shafeinejad [11] proposed an authenticated certificate-less asymmetric searchable encryption scheme for smart healthcare scenarios. In addition to resisting IKGA and Denial-of-Service (DoS) attacks, their scheme incorporates a fair payment feature and introduces the concept of a secure emergency alert. Their design, however, is based on pairing operations.

To reduce the computational overhead on medical sensors, the schemes proposed in [12, 13] eliminate the costly pairing operations. Cheng et al. [14] investigated the vulnerabilities of authenticated certificateless searchable encryption schemes, demonstrating their susceptibility to frequency analysis attacks. In such attacks, an adversary can infer the searched keywords by analyzing the frequency distribution of trapdoors and comparing it with the public frequency distribution of keywords. To counter this, Zhang et al. [15] proposed a certificateless scheme with a designated tester, focusing on thwarting frequency analysis by establishing the indistinguishability of trapdoors and ciphertexts. Other contributions include a scheme by [16] that supports range queries. Elhabob et al. [17] presented a scheme with an equality test capability for the Internet of Vehicles (IoV), allowing a cloud server to check the equality of two ciphertexts without accessing their content. Similarly, Tian et al. [18] proposed a unified scheme for search and equality testing that is resistant to message recovery attacks. Tsai and Yang [19] introduced a scheme with an equality test that is secure against side-channel attacks, ensuring security in high-risk environments through continuous key updates.

Li et al. [20] utilized proxy re-encryption (by a trusted party) to propose a scheme for smart city environments. Liu et al. [21], also using proxy re-encryption, designed

a bilinear pairing-based scheme that allows the Data Owner (DO) to dynamically add recipients by generating re-encryption keys, without needing to re-encrypt or restructure the data. Addressing privacy challenges in the IIoT, Zhou et al. [22] presented an Identity-Based Broadcast Proxy Re-encryption (IB-BPRE) scheme. This work employs polynomial coefficients and Lagrange interpolation to conceal recipient identities during the re-encryption key generation process. Consequently, the cloud service provider performs re-encryption without accessing the original content, and each recipient can decrypt the data without knowledge of other members' identities.

Guo et al. [23] proposed a lightweight framework for 5G wireless networks, using the Schnorr signature to bind the user's ID with public and secret parameters. This measure prevents public key replacement attacks and partial key leakage. Taha et al. [24] combined certificateless cryptography with cryptographic reverse firewalls to offer a solution against algorithm substitution attacks in IIoT. Their scheme is based on bilinear mappings, and its security was proven in the random oracle model.

1.1.3 Blockchain-based Schemes

Zou et al. [25] introduced a scheme supporting multi-keyword fuzzy search using the SKNN algorithm. In their architecture, attribute management is handled via tokens on a blockchain. Tan et al. [26] proposed a cloud-edge collaboration model for sharing 5G-based traffic data, utilizing Ciphertext-Policy Attribute-Based Keyword Search (CP-ABKS). The main novelty of their work is the use of a Least Recently Used (LRU) cache mechanism at edge nodes to reduce latency and the leveraging of a blockchain to ensure the integrity of indexes against semi-honest servers. Yin et al. [27] proposed a decentralized CP-ABKS scheme for blockchain-based Electronic Health Record (EHR) systems. They aimed to resolve the dependency on a central attribute authority, which creates a single point of failure (SPoF) and has access to all keys and confidential information. They solved this by distributing key shares among multiple attribute authorities on a blockchain. Li et al. [28] decentralized periodic key generation using secret sharing and consensus committees within a blockchain. Their scheme also ensures that the compromise of a single user's private key does not breach the system's security. Designed for multi-reader environments, it provides a mechanism for dynamic user revocation while maintaining constant-size ciphertexts.

Han et al. [29] proposed a blockchain-integrated public-key searchable encryption framework to address the tracing gap. Aiming for compliance with the General Data Protection Regulation (GDPR), they introduced the concept of strong traceability, which enables the simultaneous identification of both the user's identity and the searched keywords. Their architecture employs zero-knowledge proofs for blind trapdoor generation to preserve keyword confidentiality.

Jin et al. [30] proposed a scheme that uses a Cuckoo filter structure to achieve near $O(1)$ search complexity. Their model replaces the third-party auditor with smart contracts, thus eliminating the single point of failure challenge. In a subsequent work [31], they utilized a bitmap data structure for indexing, enabling high-speed multi-keyword search through bitwise logical operations. Despite the innovation in eliminating data duplication, the high on-chain gas cost reduces its efficiency for latency-sensitive applications in the IoMT domain.

Wang et al. [32] proposed a scheme that uses aggregate keys for the verification of search results and to facilitate fair interactions. Ghosh et al. [33] presented a framework for secure data sharing in remote healthcare environments. Their primary innovation is countering internal keyword guessing attacks by combining private and consortium blockchains.

1.2 Motivations

While recent works such as [11] have successfully integrated blockchain with Searchable Encryption (SE) to ensure data integrity and transparency, they exhibit several critical limitations that hinder their practical deployment in Internet of Medical Things (IoMT) environments. First, the underlying cryptographic construction in [11] relies heavily on computationally expensive bilinear pairings, which is highly inefficient for resource-constrained IoMT devices. Second, executing search operations directly via smart contracts introduces severe scalability bottlenecks and exorbitant gas costs. Finally, their scheme lacks support for advanced search functionalities. In real-world e-health scenarios, capabilities such as range queries (e.g., retrieving patient records with blood glucose levels between 140 and 180 mg/dL) are essential. Overcoming these computational, architectural, and functional shortcomings is the primary motivation of this paper.

1.3 Contributions

To address the limitations of existing frameworks, particularly the scheme proposed in [11], this paper introduces a novel, lightweight, and scalable IoMT data management framework. Our main contributions are summarized as follows:

- **Pairing-Free Searchable Encryption:** We design a lightweight, certificate-less asymmetric SE scheme that completely eliminates the need for expensive pairing operations. This significantly reduces the computational overhead for index and trapdoor generation compared to the pairing-based approach in [11], making it highly efficient and practical for IoMT devices.
- **Scalable Blockchain Architecture:** Unlike [11], which executes searches directly on-chain, our framework delegates the search execution to the Cloud Service Provider (CSP). We introduce a data digest roll-up mechanism where the smart contract acts merely as an immutable verifier for public auditing, effectively resolving the blockchain scalability and storage bottlenecks.
- **Advanced Search Functionalities:** We expand the functional capabilities of the system by natively supporting efficient range queries and conjunctive searches. This addresses a major functional gap in [11] and makes the proposed scheme suitable for complex medical data retrieval.

2 Preliminaries

2.1 0/1-encoding

In this scheme, the forward ciphertext retrieval function utilizes a 0/1-encoding method [34]. An n -bit binary integer $X = x_1x_2 \dots x_n \in \{0, 1\}^n$ is mapped into two distinct sets via the following algorithms:

- **0-encoding:**

$$S_X^0 = \{x_1x_2 \dots x_i1 \mid x_{i+1} = 0, 1 \leq i < n\}$$

- **1-encoding:**

$$S_X^1 = \{x_1x_2 \dots x_i \mid x_i = 1, 1 \leq i \leq n\}$$

Set Cardinality: If X is a non-zero n -bit string, the maximum cardinality of both S_X^0 and S_X^1 is strictly bounded by $n - 1$.

Key Property: For any two integers X and Y , the following condition holds:

$$X < Y \iff S_X^0 \cap S_Y^1 \neq \emptyset$$

Rationale: This property holds because each element in the 0-encoding represents a prefix of X terminating at an index where the next bit in X is 0 (encoded as 1 in S_X^0). An intersection with S_Y^1 signifies that Y shares the exact same prefix but contains a 1 at the specific index where X contains a 0. This bitwise divergence at the most significant differing bit mathematically guarantees that $X < Y$.

2.2 Merkle Tree

Merkle Tree [35] is a cryptographic data structure widely used to verify data integrity and efficiently manage large datasets, for which we define the root node generation algorithm as $MKTreeRoot()$. The tree is constructed by partitioning the dataset into input data chunks $D = \{d_1, d_2, \dots, d_n\}$, where each chunk d_i is hashed using a cryptographic hash function $H(\cdot)$ to generate the leaf nodes $L_i = H(d_i)$ for $i = 1, 2, \dots, n$; parent nodes are then generated by concatenating and hashing pairs of adjacent child nodes as $P_j = H(L_{2j-1} \parallel L_{2j})$, where $\parallel$ denotes concatenation, and this iterative hashing process continues layer by layer, with each level k producing nodes according to $Level_{k+1}[j] = H(Level_k[2j-1] \parallel Level_k[2j])$, while if a layer contains an odd number of nodes, the last node is duplicated and hashed with itself as $P_j = H(L_{2j-1} \parallel L_{2j-1})$, until a single root hash is obtained as $R = MKTreeRoot(D) = Level_{final}[1]$, which serves as a compact cryptographic commitment to the entire dataset enabling rapid verification of data integrity; a key efficiency property of this structure is that when a single input data chunk is modified, only the path from the affected leaf to the root requires recomputation rather than rebuilding the entire tree, so that since the tree height is $\log_2(n)$ for n chunks, this update operation has computational complexity $O(\log n)$ instead of $O(n)$, and in distributed systems Merkle Trees facilitate rapid detection and synchronization of data differences across nodes, while in blockchain and cryptocurrency applications they play a fundamental role in ensuring the integrity and security of transaction records.

Table 1: Notations used in system modeling

Symbol	Description
CSP	Cloud Service Provider
SC	Smart Contract
DO	Data Owner
DU	Data User
gp	Global parameters
sk_U	Self-generated private key of $U \in \{DO, DU\}$
pk_U	Self-generated public key of $U \in \{DO, DU\}$
β_U	Hashed identity of user $U \in \{DO, DU\}$
I_w	Index of category w
$\mathcal{K}$	Session key
$\mathcal{H}$	Shared key between DO and DU
δ	MAC chain value for category c
C_{I_c}	Counter for category c
θ_ψ	MAC chain value for pair (v, c)
$\mathcal{C}_\psi$	Counter for pair (v, c)
r_i	Range of values for category c_i
$s^\mathcal{T}$	Start date
$e^\mathcal{T}$	End date
$\mathcal{T}_{(c,v)}$	Trapdoor of pair (c, v)
T_w	Trapdoor of category w
Δ	MAC chain value for category c in results
Θ	MAC chain value for pair (c, v) in results
C_1	Counter for category c in results
C_2	Counter for pair (c, v) in results
$\mathcal{H}$	Current document checkpoint
$\mathcal{H}$	Previous document checkpoint

3 The proposed framework

3.1 Notations

Table 1 Summarizes the definition of symbols used in throughout of the paper.

3.2 System model

The system model of the proposed framework comprises four primary components:

- The DO, representing the patient, encrypts documents and stores them in the CSP. The scheme supports multiple DOs.
- The DU, representing the physician, searches documents using keywords, downloads the desired documents, and decrypts them. The scheme supports a single DU.
- The CSP is responsible for storing and retrieving documents. The scheme supports multiple CSPs.
- The SC, deployed by the DU, facilitates interactions between the DO and the DU.

3.3 Threat model

The system design has four main components, each with unique security features. The DO is honest but curious and may try to collect others' private data. The DU is also honest but curious and may seek unauthorized data, like messages between patients and other physicians. The blockchain infrastructure, due to its features, acts as a

trusted component. The CSP is dishonest and curious, seeking access to DOs' data and possibly compromising the integrity of the results. The CSP's malicious functions are as follows:

- Return incomplete results
- Delete some documents
- Reordering results
- Injecting fabricated results
- Duplicate documents
- Return results which do not correspond with DU queries
- Alter documents or other uploaded data
- Violation of confidentiality

3.4 Design Considerations

The proposed framework is built around the following design components:

- **MAC-Chain Integrity Mechanism:** Data integrity is maintained through a MAC-chain mechanism integrated with blockchain, operating across three phases: (1) *Upload*, in which indices, MAC chain values, and counters are computed for document categories, local memory is updated, and identifiers along with encrypted content are transmitted to the CSP; (2) *Checkpoint*, in which MAC chain values and counters for all categories from interval documents are recorded on the SC; and (3) *Verification*, in which data validity is confirmed by recomputing MAC chain values and comparing them against blockchain records.
- **Blockchain and Smart Contract Integration:** The blockchain serves as an immutable ledger for checkpoint data, including cr , counters, MAC chain values, and the legitimate MAC key. When the DO discloses the key, public auditing is enabled, allowing any party to independently recompute MAC chain values and counters for comparison with on-chain records. The DU, who owns the SC, can invoke specialized smart contract methods—such as activation, deactivation, and DO registration—to manage system interactions. The SC also exposes read and write interfaces for checkpoint data.
- **Searchable Encryption with Formal Security:** The SE scheme protects DO data confidentiality against both external adversaries (e.g., the CSP) and internal threats (e.g., other DOs or DUs) by securing indices and trapdoors. The proposed scheme addresses index indistinguishability, unauthorized trapdoor generation, and keyword guessing attacks (KGAs).
- **Pairing-Free Design and Extended Search Capabilities:** To reduce computational overhead, the scheme adopts a pairing-free construction that accelerates index generation and trapdoor creation. The test algorithm operates in $O(1)$ time complexity. Building on this efficiency, the framework supports range queries, which generalize naturally to exact-match and conjunctive search operations. Data is stored redundantly across multiple CSPs, ensuring availability through a distributed architecture.

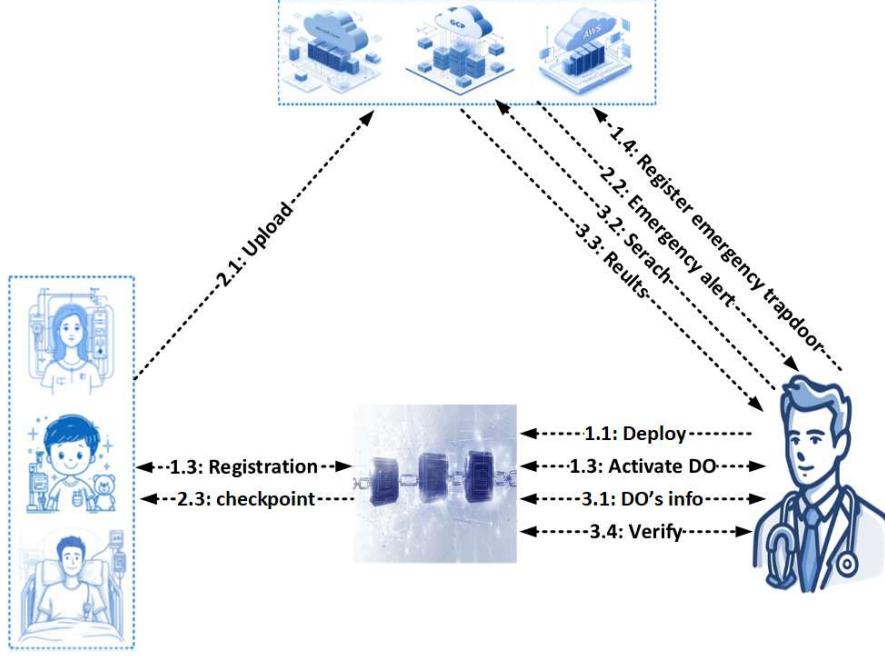


Fig. 1: Overview of system model for the proposed scheme

3.5 Framework architecture

Figure 1 illustrate proposed framework architecture. The proposed framework includes three phases:

3.5.1 Setup

The DU Initiation mechanism forms the base for starting a cryptographic protocol. It starts by creating pairs of public and private keys for all users (DOs and DUs) using set global values. The protocol is based on the Diffie-Hellman Assumption. After generating keys, the DU sets up an SC on the blockchain. The SC acts as a proxy between DOs and DUs, blocking attacks such as Denial of Service (DoS) and replay attacks, and stores hashes of messages from DOs to DUs. The DU then publishes its public key with the address of the SC.

Algorithm 1 presents the algorithm for the DO Initialization Process. This process implements a secure registration and indexing framework for DOs. The process begins with the DO registering its credentials and public key on the SC. For each category c in the dictionary, the algorithm constructs a primary index I_c and initializes (δ_{I_c}, C_{I_c}) to zero. For each category-value pair (c, v) , a composite index ψ is constructed by hashing the concatenation of primary indices. (θ_ψ, C_ψ) are initialized to zero. The algorithm generates a random key $\mathcal{K}[L]$ and applies recursive hashing to derive L previous keys, thereby creating a cryptographic key chain. After initializing $\mathcal{P}$ and $\mathcal{P}$, the algorithm returns $(\{I_c\}, \{\psi\}, \mathcal{K}[0..L], \mathcal{P}, \mathcal{P})$ as the DO's initialized state. The

Algorithm 1 DO Initialization Process

Require: $gp, sk_{DO}, pk_{DO}, dict, name, age$

Ensure: $(\{I_c\}, \{\psi\}, \mathcal{K}[0..L], \mathcal{P}, \mathcal{P})$

```
1: INVOKE register( $name, age, pk_{DO}$ ) on SC
2: for all  $c \in dict.categories$  do
3:    $I_c \leftarrow \text{BuildIndex}(c, pk_{DU}, sk_{DO}, gp)$ 
4:    $(\delta_{I_c}, C_{I_c}) \leftarrow (0, 0)$ 
5:   for all  $v \in dict[c]$  do
6:      $\mathcal{I}_v \leftarrow \text{BuildIndex}(v, pk_{DU}, sk_{DO}, gp)$ 
7:      $\psi \leftarrow \text{Hash}(I_c \parallel \mathcal{I}_v)$ 
8:      $(\theta_\psi, \mathcal{C}_\psi) \leftarrow (0, 0)$ 
9:   end for
10: end for
11:  $\mathcal{K}[L] \leftarrow \text{Random} \in \{0, 1\}^\lambda$ 
12: for  $i \leftarrow L - 1$  downto 0 do
13:    $\mathcal{K}[i] \leftarrow \text{Hash}(\mathcal{K}[i + 1])$ 
14: end for
15:  $\mathcal{P} \leftarrow \emptyset$ 
16:  $\mathcal{P} \leftarrow \emptyset$ 
17: return  $(\{I_c\}, \{\psi\}, \mathcal{K}[0..L], \mathcal{P}, \mathcal{P})$ 
```

time complexity is $O(\text{dict.size} + L)$, where dict.size is the number of category-value pairs in the dictionary, and L is the number of checkpoints required for the system's operation.

3.5.2 Storing

[Algorithm 2](#) sets up a secure process for immediate document upload and indexing, ensuring confidentiality, integrity, and authenticity. The protocol first uses a symmetric key ($\mathcal{K}$) to encrypt documents, creating ciphertext (ξ_{Doc_i}). This ciphertext is hashed to make a unique document ID. Each category and each (category, value) pair is handled in turn. Indexes for these pairs use an asymmetric searchable encryption method with the DU's public key (pk_{DU}) and the DO's private key (sk_{DO}). Counters are updated, and HMAC chain tags ($\delta_{I_c}, \theta_\psi$) are created to link indexes, counters ($C_{I_c}, \mathcal{C}_\psi$), and document IDs for each category and (category, value) pair. $\delta_{I_c}, \theta_\psi$ are HMAC chain tags for a category and (category, value) pair. This allows modification tracking and protects integrity. The DO keeps these records locally. The encrypted document and its metadata are then uploaded to the CSP, balancing searchability with confidentiality and integrity. The algorithm runs in $O(\text{doc.pairs.size}) * (\text{order}(\text{BuildIndex}) + \text{order}(\text{HMAC}))$ time, where doc.pairs is the count of category-value pairs in the document.

3.5.3 Checkpoint

[Algorithm 3](#) outlines a procedure for aggregating uploaded data within the current checkpoint interval and storing the resulting digest values on the SC. The algorithm

Algorithm 2 Immediate Document Upload and Indexing

Require: $Doc_i, \mathcal{H}, pk_{DU}, sk_{DO}, \mathcal{K}_{cr}, cr, gp$ **Ensure:** $\mathcal{K}_{cr}$

```
1:  $\xi \leftarrow Enc(\mathcal{H}, Doc_i)$ 
2:  $\mathcal{ID} \leftarrow H(\xi)$ 
3:  $\mathcal{S} \leftarrow \emptyset, \mathcal{I} \leftarrow \emptyset$ 
4: for all  $(c, v) \in Doc_i$  do
5:    $I_c \leftarrow Idx(c, pk_{DU}, sk_{DO}, gp)$ 
6:    $\mathcal{I} \leftarrow \mathcal{I} \cup \{I_c\}$ 
7:    $\mathcal{I}_v \leftarrow Idx(v, pk_{DU}, sk_{DO}, gp)$ 
8:    $\psi \leftarrow H(I_c, \mathcal{I}_v)$ 
9:    $\mathcal{S} \leftarrow \mathcal{S} \cup \{\psi\}$ 
10:   $C_{I_c} \leftarrow C_{I_c} + 1$ 
11:   $\delta_{I_c} \leftarrow HMAC(\mathcal{K}_{cr}, (\mathcal{ID}, \delta_{I_c}, I_c, C_{I_c}))$ 
12:   $\mathcal{C}_\psi \leftarrow \mathcal{C}_\psi + 1$ 
13:   $\theta_\psi \leftarrow HMAC(\mathcal{K}_{cr}, (\mathcal{ID}, \theta_\psi, \psi, \mathcal{C}_\psi))$ 
14:  DO updates locally  $(\delta_{I_c}, C_{I_c}, \theta_\psi, \mathcal{C}_\psi)$ 
15: end for
16: DO uploads  $(\xi, \mathcal{S}, \mathcal{I}, \{\delta_{I_c}\}, \{C_{I_c}\}, \{\theta_\psi\}, \{\mathcal{C}_\psi\}, cr)$  to CSP
17: return  $\mathcal{K}_{cr}$ 
```

Algorithm 3 Checkpoint

Require: $gp, cr, \mathcal{K}_{cr}$ **Ensure:** –

```
1:  $\mathcal{P} \leftarrow$  retrieve set of updated keys in this interval
2:  $\mathcal{P} \leftarrow$  retrieve dictionary of updated pairs in this interval
3:  $\mathcal{U} \leftarrow \emptyset$ 
4: for all  $c \in \mathcal{P}$  do
5:    $\mathcal{U} \leftarrow \emptyset$ 
6:   for all  $v \in \mathcal{P}[c]$  do
7:      $\mathcal{U} \leftarrow \mathcal{U} \cup \{(\theta_\psi, \mathcal{C}_\psi)\}$ 
8:   end for
9:    $\mathcal{U} \leftarrow \mathcal{U} \cup \{(\delta_{I_c}, C_{I_c}, \mathcal{U})\}$ 
10: end for
11: Invoke  $writeData(cr, \mathcal{U}, key = \mathcal{K}_{cr})$  on SC
```

consolidates cryptographic evidence of modifications during the interval, thereby generating verifiable blockchain records. It retrieves the updated categories ($\mathcal{P}$) and the updated pairs ($\mathcal{P}$). Then, it fetches the HMAC chain tags and counters used for data integrity and sequencing. The SC function is subsequently invoked to permanently record these values, ensuring both confidentiality and verifiability. The time complexity of this algorithm is $O(\mathcal{H}.updatedPairs.size)$, where $\mathcal{H}.updatedPairs$ represents the number of category-value pairs updated during the checkpoint $\mathcal{H}$ interval.

Algorithm 4 DU-Search Function

Require: $gp, sk_{DU}, pk_{DO}, (v_1, r_1), \dots, (c_n, r_n), s^T, e^T$ **Ensure:** $\mathcal{R}$

```
1:  $\mathcal{R} \leftarrow \emptyset$ 
2: for  $i \leftarrow 1$  to  $n$  do
3:    $T_c \leftarrow \text{Trapdoor}(c_i, sk_{DU}, pk_{DO}, gp)$ 
4:    $T_v \leftarrow \text{Trapdoor}(r_i, sk_{DU}, pk_{DO}, gp)$ 
5:    $\mathcal{T}_{(c,v)} \leftarrow \text{Hash}(T_c, T_v)$ 
6:    $\mathcal{D} \leftarrow \text{CSP-Search}(gp, \mathcal{T}_{(c,v)}, s^T, e^T)$ 
7:    $\mathcal{R} \leftarrow \mathcal{R} \cup \{((T_c, \mathcal{T}_{(c,v)}), \mathcal{D})\}$ 
8: end for
9: return  $\mathcal{R}$ 
```

3.5.4 Retrieval

Algorithm 4 allows the DU to perform exact match and range queries over encrypted repositories while preserving data confidentiality. The algorithm processes pairs of categories and value ranges $((c_1, r_1), \dots, (c_n, r_n))$ under temporal constraints. For each category c_i , a trapdoor T_c is generated, and for each value v , a trapdoor T_v is produced using sk_{DU} and pk_{DO} , ensuring that only authorized queries are possible. A composite token $\mathcal{T}_{(c,v)}$ is then computed via cryptographic hashing and submitted to the CSP Side Search, which aggregates document references without revealing category or value information. Temporal constraints (s^T, e^T) restrict the results to specified intervals. The algorithm has a computational complexity of $O(\mathcal{H}. \text{updatedPairs.size})$, where $\mathcal{H}$ is a hash table. updatedPairs denotes the updated category-value pairs within checkpoint $\mathcal{H}$.

Algorithm 5 checks each candidate document's cryptographic index set ($\mathcal{S}$) using the Test function, which determines if certain index entries ($I_{(c,v)}$) match the search trapdoor without revealing information about corresponding values. The oblivious matching process allows CSP to precisely determine matched documents while preserving document confidentiality. Therefore, we achieved a balance between privacy protection and computational performance. For $\mathcal{H}. \text{updatedPairs}$, representing the number of category-value pairs updated during the checkpoint $\mathcal{H}$ interval, the time complexity of this technique is $O(\mathcal{H}. \text{updatedPairs.size})$.

A cryptographic verification mechanism is presented by Algorithm 6 to ensure the integrity of the results that CSP returns without sacrificing secrecy. In order to create periodic verification chains for blockchain checkpoint validation, the protocol first processes search trapdoors $(T_c, \mathcal{T}_{(c,v)})$ and a proof chain ($\mathcal{PT}$) generated by documents and cryptographic proof. For each document, it conducts hierarchical validations: verifying document integrity by matching identifiers to hashes of encrypted content; validating blockchain states via retrieved checkpoint records ($\mathcal{H}$) anchored to immutable states; confirming key chain relationships; and applying recursive HMAC-based accumulated authentication to detect reordering attacks. Using distinct chains with counter-based sequencing to prevent replay attacks, the procedure changes based on the kind of verification (attribute-value pairs or attributes alone). This technique

Algorithm 5 CSP-Search Function

Require: $gp, \mathcal{T}_{(c,v)}, s^T, e^T$ **Ensure:** $\mathcal{R}$

```
1:  $\mathcal{D} \leftarrow$  all stored documents in CSP uploaded between  $s^T$  and  $e^T$ 
2:  $\mathcal{R} \leftarrow \emptyset$ 
3: for all  $doc \in \mathcal{D}$  do
4:   for all  $I_{(c,v)} \in doc.\mathcal{S}$  do
5:     if  $Test(gp, \mathcal{T}_{(c,v)}, I_{(c,v)})$  then
6:        $\mathcal{R} \leftarrow \mathcal{R} \cup \{doc\}$ 
7:     end if
8:   end for
9: end for
10: return  $\mathcal{R}$ 
```

promotes verifiable computing in encrypted document systems by providing trustless validation while maintaining secrecy. It is equipped with exact error classification for detecting violations such as tampering and incompleteness.

Algorithm 7 presents a secure document retrieval method that integrates search, verification, and decryption processes. The protocol ensures that retrieved documents meet both search criteria and integrity requirements before granting plaintext access. Utilizing credentials (sk_{DU}, pk_{DO}) and specified parameters, the protocol adopts a verification-first approach, initiating with the Document User search function (DU-Search). Blockchain-based proofs are employed to validate search results, and decryption is performed only after successful verification. The results are systematically collected to maintain the association between search criteria and plaintext content. The symmetric key $(\mathcal{K})$ used for decryption operates independently of the search and verification processes, thereby mitigating confidentiality risks even if search functions are compromised. This approach ensures data integrity, completeness, and authenticity without disclosing sensitive information, making it well-suited for deployment in untrusted cloud environments. The time complexity is $O(\mathcal{H} \cdot \text{updated-Pairs.size})$, where the category-value pairs updated at checkpoint $\mathcal{H}$ are denoted as Pairs.

3.6 Proposed scheme

3.6.1 GlobalSetup(λ)

Given a security parameter λ , the trusted authority initializes the system by selecting a cyclic additive group G of prime order q with generator P . Three cryptographic hash functions are defined as $H : G \rightarrow \{0,1\}^l$ for some positive integer l , $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The authority further chooses a master secret key $msk \in \mathbb{Z}_q^*$ uniformly at random. The global public parameters are then published as $gp = \{G, P, q, H, H_1, H_2\}$.

3.6.2 IDGen(gp, msk)

This algorithm is executed by the Key Generation Center (KGC), which is modeled as an honest-but-curious entity. For each user U with identity string ID_U , the KGC samples $r_U \xleftarrow{\$} \mathbb{Z}_q^*$ and computes:

$$\alpha_U = r_U \cdot msk, \quad (1)$$

$$\beta_U = (r_U + H_1(ID_U)) \cdot P. \quad (2)$$

The output is the identity credential (α_U, β_U) , which binds the user's identity to cryptographic material without requiring certificates. α_U is private and β_U is public

3.6.3 KeyGen(gp)

To prevent the KGC from gaining full control of users' secret keys, each user independently generates a local secret/public key pair. First he/she select $x_U \xleftarrow{\$} \mathbb{Z}_q^*$ randomly. then corresponding keys are set as: $(sk_U, pk_U) = (x_U, x_U \cdot P)$. This ensures that the effective private key is distributed between the KGC and the user, thereby mitigating key escrow.

3.6.4 BuildIndex($w_i, pk_{DU}, sk_{DO}, gp$)

To enable keyword search, DO generates an index for each keyword w_i targeted to DU with public key pk_{DU} . Using its secret key sk_{DO} and identity credential, the DO computes: $I_i = H(sk_{DO} \cdot H_2(w_i) \cdot pk_{DU} + \alpha_{DO}(\beta_{DU} - H_1(ID_{DU}) \cdot P))$. The resulting index I_i is uploaded alongside the encrypted data.

To support range queries, the target values must first be normalized. For instance, considering body temperature data with a single decimal precision within an allowable range of $[13, 50]$, the normalization of the value 37.7 proceeds as follows: it is multiplied by 10 to yield 377, followed by the subtraction of 130 (i.e., 10×13 , the scaled lower bound). To ensure the result is strictly non-zero, 1 is added, yielding 248. Under this formulation, the maximum allowable value becomes 371, which requires a 9-bit representation. Given that no elements within the range contain exclusively all zeros or all ones in their bit representation, the maximum cardinality of both encoding-0 and encoding-1 sets is 8. Similarly, extending the precision to two decimal places involves multiplication by 100, leading to a maximum range value of 3701. This requires 12 bits for representation, capping the cardinality of the encoding-0 and encoding-1 sets at 11. Following normalization, the encoding-1 set of the target value is computed, and an index is generated for each of its constituent members.

3.6.5 Trapdoor(w, pk_{DO}, sk_{DU}, gp)

To perform a search, the DU generates a trapdoor for the query keyword w by combining its private key sk_{DU} with the public key of the DO. The trapdoor is computed as: $T = H(sk_{DU} \cdot H_2(w) \cdot pk_{DO} + \alpha_{DU}(\beta_{DO} - H_1(ID_{DO}) \cdot P))$. This trapdoor preserves keyword privacy while enabling equality testing over encrypted data.

To retrieve all documents containing values strictly greater than the query value, the encoding-0 set of the query value is computed, and a trapdoor is generated for each of its elements. If any of these trapdoors matches the generated indexes (denoted as $S_T^0 \cap S_I^1 \neq \emptyset$) it implies that the query value T is strictly less than the document's value I . Conversely, to retrieve documents with values less than the query value, the encoding-0 set is computed and subjected to the same process, but the system returns documents that yield no matches. A generic range query is achieved by combining both aforementioned strategies. For conjunctive queries, trapdoors are generated for all queried classes. Subsequently, trapdoors for all corresponding values are computed, sorted according to their respective classes, and aggregated into a root value via the $MKTreeRoot(\cdot)$ function. Finally, the class trapdoors, alongside the computed $MKTreeRoot$ for the value trapdoors, are dispatched to the server.

3.6.6 Test(gp, I_i, T)

For exact match and range queries, the server evaluates the presence or absence of trapdoor matches and returns the result accordingly (by checking $I_i = T$). When indexes are stored in a hash table, this operation can be performed in expected constant time, i.e., $O(1)$, thereby ensuring efficient keyword search. For conjunctive searches, the server verifies the matching of class trapdoors against the stored indexes for each document. Upon identifying the values corresponding to the queried trapdoors, it dynamically computes their $MKTreeRoot$. Ultimately, if the computed root perfectly matches the submitted root value, the conjunction is satisfied, and a successful match is established.

3.6.7 correctness

For any honestly generated keys and identities, the proposed scheme satisfies perfect completeness, i.e., for $w = w_i$ the test always outputs 1, and for $w \neq w_i$ equality $I_i = T$ holds only with negligible probability.

Proof we assume $T_1 = sk_{DU} \cdot H_2(w) \cdot pk_{DO}$ and $T_2 = \alpha_{DU}(\beta_{DO} - H_1(ID_{DO}) \cdot P)$. Therefor $T = H(T_1 + T_2)$. Also we assume $I_{i1} = sk_{DO} \cdot H_2(w_i) \cdot pk_{DU}$ and $I_{i2} = \alpha_{DO}(\beta_{DU} - H_1(ID_{DU}) \cdot P)$. Therefor $I_i = H(I_{i1} + I_{i2})$:

$$T_1 = sk_{DU} \cdot H_2(w) \cdot pk_{DO} = sk_{DU} \cdot sk_{DO} \cdot P \cdot H_2(w) = pk_{DU} \cdot sk_{DO} \cdot H_2(w).$$

$$\begin{aligned} T_2 &= \alpha_{DU}(\beta_{DO} - H_1(ID_{DO}) \cdot P) = r_{DU} \cdot msk((r_{DO} + H_1(ID_{DO})) \cdot P - H_1(ID_{DO}) \cdot P) \\ &= r_{DU} \cdot msk \cdot r_{DO} \cdot P = \alpha_{DO} \cdot (r_{DU} \cdot P) = \alpha_{DO} \cdot (r_{DU} \cdot P + H_1(ID_{DU}) \cdot P - H_1(ID_{DU}) \cdot P) \\ &= \alpha_{DO} \cdot (\beta_{DU} - H_1(ID_{DU}) \cdot P) = I_{i2} \end{aligned}$$

when $w = w_i$:

$$T_1 = pk_{DU} \cdot sk_{DO} \cdot H_2(w_i) = I_{i1} \Rightarrow T_1 + T_2 = I_{i1} + I_{i2} \Rightarrow H(T_1 + T_2) = H(I_{i1} + I_{i2}) \Rightarrow T = I_i$$

when $w \neq w_i$:

$$\text{We can say } H_2(w) \neq H_2(w_i) \text{ (otherwise we find a collision on } H_2\text{): } T_1 \neq I_{i1} \Rightarrow T_1 + T_2 \neq I_{i1} + I_{i2} \Rightarrow H(T_1 + T_2) \neq H(I_{i1} + I_{i2}) \Rightarrow T \neq I_i$$

□

3.7 Smart contract

The proposed smart contract utilizes structured data types and administrative functions to securely and transparently manage DOs. Attributes such as `name`, `age`, `publicKey`, and verification status (`verified`) are encapsulated within the `DOInfo` struct. For scalable data modeling, `UPStruct` and `UStruct` define application parameters (`theta`, `ctr`), `delta`, and `C`, supporting recursion through arrays of `UPStruct`. Mappings, including DO address associations, `cr` to `K`, and serialized datasets for efficient storage, are managed by the `SC` struct. Core functions include `register` for initializing unverified DO entries, `verifyDO` and `deactivateDO` for administrator (DU) management under `onlyAdmin` restrictions, `writeData` for securely encoding and storing hierarchical arrays with keys and emitting auditable events, and `readData` for transparent retrieval by authorized parties. This design ensures security, scalability, and auditability in decentralized data management.

- **register:** Initializes a DO with `name`, `age`, and `publicKey`, setting `verified` to false.
- **verifyDO:** Allows the DU, under the `onlyAdmin` restriction, to set a DO's `verified` status to true.
- **deactivateDO:** Permits the DU to revert a DO's `verified` status to false, thereby revoking their access rights.
- **writeData:** Serializes an array of `UStruct` instances using `abi.encode`, associates the result with a key `K`, and emits a `DataWritten` event containing `cr` and the serialized data.
- **readData:** Retrieves the key `K` and serialized data for a specified `cr`. This function is publicly accessible to enable auditing.

The `onlyAdmin` modifier allows only the DU to access admin functions, while the `DataWritten` event enables off-chain monitoring of data updates. This architecture uses blockchain immutability and transparency for secure, verifiable data management.

```
1  pragma solidity ^0.8.0;
2
3  contract se {
4      struct DOInfo { string name; uint age; bytes publicKey; bool verified; }
5      struct UPStruct { uint256 theta; uint256 ctr; }
6      struct UStruct { uint256 delta; uint256 C; UPStruct[] UP; }
7
8      address public DU;
9      mapping(address => DOInfo) public DOs;
10     mapping(uint256 => uint256) public dataKeys;
11     mapping(uint256 => bytes) public serializedData;
12
13     constructor() { DU = msg.sender; }
14
15     modifier onlyAdmin() {
16         require(msg.sender == DU, "Only admin");
17         _;
18     }
19
20     event DataWritten(uint256 indexed cr, uint256 K, bytes uData);
21 }
```

```

22     function register(string memory _name, uint _age, bytes memory _pk) public {
23         DOs[msg.sender] = DOInfo(_name, _age, _pk, false);
24     }
25
26     function verifyDO(address _addr) public onlyAdmin {
27         require(!DOs[_addr].verified, "Activated");
28         DOs[_addr].verified = true;
29     }
30
31     function deactivateDO(address _addr) public onlyAdmin {
32         require(DOs[_addr].verified, "Not active");
33         DOs[_addr].verified = false;
34     }
35
36     function writeData(uint256 cr, UStruct[] calldata U, uint256 K) public {
37         dataKeys[cr] = K;
38         serializedData[cr] = abi.encode(U);
39         emit DataWritten(cr, K, serializedData[cr]);
40     }
41
42     function readData(uint256 cr) public view returns (uint256 K, bytes memory data) {
43         return (dataKeys[cr], serializedData[cr]);
44     }
45 }

```

Listing 1: Smart contract implementation

4 Security analysis

The proposed CL-ASE scheme is provably secure, as established through formal game-based proofs. Building on this, the analysis demonstrates that, under the Decisional Diffie-Hellman assumption and the one-wayness of the employed hash functions, the scheme achieves keyword confidentiality, trapdoor privacy, and resistance to IKGA. Furthermore, security is established through reductions, which ensure that any adversary capable of breaking the scheme can be transformed into an algorithm that solves an underlying hard problem with non-negligible probability.

4.1 Scheme security

Theorem 1 *Assume that the Computational Diffie-Hellman (CDH) problem is hard in G and that H , H_1 , and H_2 are random oracles, where $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. Let $\mathcal{A}$ be a PPT adversary that makes at most q_H queries to H and has advantage ϵ in distinguishing the index of one of two challenge keywords for the challenge pair (DO^*, DU^*) without making trapdoor queries for this challenge pair. Then there exists a PPT algorithm $\mathcal{B}$ that solves the CDH problem in G with probability at least*

$$\text{Adv}_{\mathcal{B}}^{\text{CDH}} \geq \frac{2\epsilon}{q_H}.$$

Proof Let $(P, A = aP, B = bP)$ be a challenge instance of the CDH problem in G , where $a, b \in \mathbb{Z}_q^*$ are unknown to $\mathcal{B}$. The goal of $\mathcal{B}$ is to compute abP . $\mathcal{B}$ runs $\mathcal{A}$ by simulating its challenger and the random oracles as follows:

Setup.

$\mathcal{B}$ sets the public parameters of the system. It chooses $msk \in \mathbb{Z}_q^*$ uniformly at random. For the challenge data owner DO^* and data user DU^* , $\mathcal{B}$ sets their public keys as:

$$pk_{DO^*} = A = aP \quad \text{and} \quad pk_{DU^*} = B = bP,$$

which implicitly sets $sk_{DO^*} = a$ and $sk_{DU^*} = b$ (both unknown to $\mathcal{B}$). For the identity credentials, $\mathcal{B}$ chooses $r_{DO^*}, r_{DU^*} \in \mathbb{Z}_q^*$ uniformly at random and computes:

$$\begin{aligned} \alpha_{DO^*} &= r_{DO^*} \cdot msk, & \beta_{DO^*} &= r_{DO^*}P + H_1(ID_{DO^*})P \\ \alpha_{DU^*} &= r_{DU^*} \cdot msk, & \beta_{DU^*} &= r_{DU^*}P + H_1(ID_{DU^*})P \end{aligned}$$

Since $\mathcal{B}$ knows msk , these credentials are representation-wise valid and indistinguishable from real ones. $\mathcal{B}$ defines the static offset:

$$Z = r_{DO^*} \cdot r_{DU^*} \cdot msk \cdot P,$$

which is computable since msk, r_{DO^*}, r_{DU^*} are known. For any keyword w , the correct hash input for the challenge pair is $X_w = H_2(w)abP + Z$. For any other user $U \neq \{DO^*, DU^*\}$, $\mathcal{B}$ generates their private keys $x_U \in \mathbb{Z}_q^*$ honestly and defines $pk_U = x_U P$, keeping the private keys in a local table.

Oracle Queries.

$\mathcal{B}$ maintains tables for the random oracles H_1, H_2 , and H to ensure consistency (lazy-sampling):

- **H_1 -Queries:** Upon receiving ID_U , if an entry exists in L_1 , $\mathcal{B}$ returns the recorded value. Otherwise, it samples $h_1 \in \mathbb{Z}_q^*$ randomly, records (ID_U, h_1) in L_1 , and returns h_1 .
- **H_2 -Queries:** Upon receiving keyword w , if an entry exists in L_2 , $\mathcal{B}$ returns the recorded value. Otherwise, it samples $h_2 \in \mathbb{Z}_q^*$ randomly, records (w, h_2) in L_2 , and returns h_2 .
- **H -Queries:** Upon receiving $X \in G$, if an entry exists in L_H , $\mathcal{B}$ returns the recorded value. Otherwise, it samples $V \in \{0, 1\}^l$ randomly, records (X, V) in L_H , and returns V .

Trapdoor Queries.

$\mathcal{A}$ can query trapdoors for any keyword w and any user pair $(DO, DU) \neq (DO^*, DU^*)$. Since $\mathcal{A}$ is restricted from querying trapdoors for (DO^*, DU^*) , at least one of the parties in any query (DO, DU) has a known private key (either x_{DO} or x_{DU} is known to $\mathcal{B}$). Thus, $\mathcal{B}$ can compute the hash input $X_w = H_2(w)x_{DO}x_{DU}P + r_{DO}r_{DU}mskP$ honestly and evaluate $H(X_w)$ using the simulated oracle H .

Challenge.

$\mathcal{A}$ outputs two challenge keywords w_0, w_1 for the target pair (DO^*, DU^*) . $\mathcal{B}$ samples a bit $b \in \{0, 1\}$ and a target ciphertext $I^* \in \{0, 1\}^l$ uniformly at random, and returns I^* as the challenge index to $\mathcal{A}$.

Extraction.

Let $X^* = H_2(w_b)abP + Z$ be the correct pre-image corresponding to the challenge index. Because I^* was chosen uniformly at random, $\mathcal{A}$ has no advantage in guessing b unless it queries $H(X^*)$. Let Ask^* be the event that $\mathcal{A}$ queries H on X^* . By standard probability analysis, the advantage of $\mathcal{A}$ is bounded by:

$$\text{Adv}(\mathcal{A}) = \left| \Pr[b' = b] - \frac{1}{2} \right| \leq \frac{1}{2} \Pr[\text{Ask}^*],$$

which implies $\Pr[\text{Ask}^*] \geq 2\epsilon$.

When $\mathcal{A}$ finishes, $\mathcal{B}$ chooses a random query X_j from the list of queries submitted to H (which contains at most q_H entries). $\mathcal{B}$ computes and outputs the candidate solution:

$$Q = H_2(w_b)^{-1} \cdot (X_j - Z).$$

If the chosen query X_j is indeed X^* , we have:

$$Q = H_2(w_b)^{-1} \cdot (H_2(w_b)abP + Z - Z) = abP,$$

which is the correct solution to the CDH problem. Since $\mathcal{B}$ selects the correct entry from L_H with probability at least $1/q_H$, its success probability is:

$$\Pr[\mathcal{B} \text{ succeeds}] \geq \Pr[\text{Ask}^*] \cdot \frac{1}{q_H} \geq \frac{2\epsilon}{q_H}.$$

This completes the proof. $\square$

4.2 Security against other attacks

This subsection explains how well the proposed framework can handle different possible attacks.

- **Return incomplete results:** Returning fewer documents than expected suggests missing records. The smart contract records the total number of documents associated with each c and v . Detecting a discrepancy triggers an **INCOMPLETE** error.
- **Delete some documents:** Deleting records is treated as a case of returning incomplete results. Detecting an **INCOMPLETE** error enables the client to request all documents again. Noncompliance confirms deletion, resulting in an **INCOMPLETE** error.
- **Reordering results:** The MAC chain is determined by the stored δ and θ parameters for each c and v . Altering the order of documents creates a mismatch between the stored MAC value, such as $\text{MAC}(\text{Doc}_2, \text{MAC}(\text{Doc}_1, \dots))$, and the computed MAC chain, such as $\text{MAC}(\text{Doc}_1, \text{MAC}(\text{Doc}_2, \dots))$. This mismatch results in a **TAMPERED** error.
- **Alter documents or uploaded data:** Modifying a record changes its MAC, and altering the MAC is detected due to the chain structure, resulting in a **TAMPERED** error. Inject fabricated results: Injecting fake records or replacing valid ones with fabricated records triggers detection by count discrepancy or by MAC chain mismatch, resulting in a **TAMPERED** error.
- **Duplicate documents:** Duplicating records is a case of fabricated result injection and is flagged as **TAMPERED**.
- **Return results inconsistent with DU queries:** Returning results tied to a different query creates a mismatch with the contract's MAC chain, triggering a **TAMPERED** error.

Table 2: Execution and transaction cost of the smart contract functions

Function	Execution cost	Transaction cost
register	118,578	141,670
verifyD0	25,126	46,558
deactivateD0	8,089	24,721
writeData	507,811	532,051

Table 3: Comparison with other searchable encryption schemes

Feature	Ours	[1]	[7]	[10]	[11]	[13]	[15]	[18]	[20]	[21]	[36]	[24]	[27]	[28]	[29]	[32]
Blockchain based	✓	×	×	×	✓	×	×	×	×	×	×	×	✓	✓	✓	✓
Emergency alert	✓	×	×	×	✓	×	×	×	×	×	×	×	×	×	×	×
IKGA resistance	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	×	✓	✓	×
Conjunctive search	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Range query	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
Result verification	✓	×	×	×	✓	×	×	×	×	×	×	×	×	×	×	✓
Public audit	✓	×	×	×	×	×	×	×	×	×	×	×	×	×	×	✓
Randomness	×	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Pairing Free	✓	×	✓	✓	×	✓	×	✓	×	×	×	×	×	✓	×	×
Certificate-less	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×	×
Authenticated	✓	×	✓	✓	✓	✓	✓	✓	✓	×	×	✓	×	✓	✓	×
Scalable blockchain	✓	-	-	-	×	-	-	-	-	-	-	-	×	×	×	×
Search authority	CSP	CSP	CSP	CSP	SC	CSP	CSP	CSP	CSP	CSP	CSP	CSP	CSP	CSP	CSP	CSP

- **Violate confidentiality:** Compromising confidentiality is prevented by the underlying searchable encryption scheme, as established in [subsection 4.1](#).

5 Evaluations

5.1 Gas evaluation

[Table 2](#) presents the Gas consumption data associated with the primary functions of the proposed smart contract, including both execution and transaction costs. This information is used to evaluate operational efficiency on the blockchain. The `verifyD0` function enables secure authentication and minimizes computational overhead. The `deactivateD0` function allows for efficient revocation of access rights. The `writeData` function consumes more gas because it requires increased data handling. These measured costs demonstrate the framework’s practicality by balancing security requirements with economic efficiency within decentralized systems.

5.2 Comparison to related work

In [Table 3](#), we compare our framework with prior work based on the following metrics:

Blockchain-based: Only a limited number of existing schemes integrate blockchain into searchable encryption. Unlike conventional designs such as [\[1, 7, 10, 13\]](#), the proposed scheme adopts a blockchain-based framework while also addressing scalability. In contrast to blockchain-enabled methods such as [\[11, 27–29, 32\]](#), it stores

only lightweight metadata and digests on-chain, which reduces storage overhead and operational cost.

Emergency alert: Among the compared schemes, emergency alert support is provided only by the proposed scheme and [11]. However, the proposed method combines this feature with a lightweight constant-time test mechanism, enabling fast identification of critical medical events and immediate notification to the physician.

IKGA resistance: Resistance against IKGA is not guaranteed in several existing schemes, including [1, 21, 27, 32, 36]. In contrast, the proposed scheme is formally designed to resist IKGA, similar to a subset of secure modern constructions, thereby providing stronger keyword privacy against malicious servers.

Search functionality: Most existing schemes support only basic keyword search. The proposed scheme extends search capability by supporting both conjunctive search and range query, which are absent in all compared methods. This gives the scheme greater flexibility for practical e-health retrieval scenarios.

Result verification and public audit: Only a few schemes consider verifiability. While [11] and [32] support result verification, public audit is provided only by [32]. The proposed scheme supports both properties simultaneously, allowing users and external auditors to verify correctness and integrity without exposing plaintext data.

Randomness: Unlike most existing searchable encryption schemes, the proposed design does not rely on randomness during search-related operations. This simplifies implementation and reduces computational overhead while preserving security requirements.

Pairing-free: Although some recent methods such as [7, 10, 13, 18, 28] avoid bilinear pairings, many others still rely on costly pairing computations. The proposed scheme is pairing-free and therefore achieves better practicality for resource-constrained environments.

Certificate-less and authenticated: The proposed scheme satisfies both certificate-less and authenticated design requirements. While several recent schemes provide one or both of these properties, older and blockchain-oriented alternatives such as [1, 27, 32] do not offer the same level of integrated authentication and certificate management efficiency.

Scalable blockchain: Among blockchain-based schemes, scalability remains a major limitation. Existing blockchain-enabled methods such as [11, 27–29, 32] do not provide a scalable blockchain design. In contrast, the proposed scheme explicitly incorporates a scalable blockchain strategy, improving its suitability for large-scale deployment.

Search authority: In most compared schemes, the search task is handled by the CSP. A notable exception is [11], which delegates this function to the smart contract. The proposed scheme assigns the search operation to the CSP, which reduces on-chain cost while still preserving verifiability and security.

5.3 Experimental analysis

For the experimental analysis, the proposed searchable encryption scheme was implemented using the Apache Milagro Crypto Library (AMCL) from MIRACL Labs in the C programming language. The experiments used a Windows 11 system (64-bit) with

a 12th Gen Intel(R) Core(TM) i5-12400 2.50 GHz CPU and 16GB of RAM. Figure 2a shows the BuildIndex time for 50 to 300 keywords. The proposed scheme consistently achieves the lowest execution time, followed by [13], while [11], [24], and [36] impose higher overhead, with [36] being the slowest. For large-scale search (Figure 2b), both the proposed scheme and [11] preserve nearly constant execution time as the keyword set grows, whereas the other methods increase linearly. In addition, Figure 2c presents the execution time of range query trapdoor generation for intervals increasing from 10 to 90 values, and Figure 2d reports the corresponding range query search time over data groups of ten documents. In both cases, the proposed scheme shows stable and efficient performance, while the execution time of the competing schemes rises with the query range. These results demonstrate the effectiveness and scalability of the proposed searchable encryption scheme.

In Table 5, the notation $|G_{AG}|$ denotes the size of an element in the additive group, $|G_{MG}|$ denotes the size of an element in the multiplicative group, $|Z_p|$ denotes the size of an element in Z_p , and v denotes the number of keywords. Furthermore, in both Table 4 and Table 5, the components highlighted in red for certain schemes correspond to the re-encryption operation; since neither our proposed scheme nor some of the other listed schemes perform re-encryption, these components are excluded from the comparison to ensure a fair evaluation. As summarized in Table 4 and Table 5, the computational and communication overhead of our scheme scales favorably with query type. Table 5 For a conjunctive search over t fields, the communication complexity of the trapdoor generation phase in our scheme is multiplied by a factor of $(t+1)$, whereas in the competing schemes listed in Table 5 the corresponding complexity is multiplied by a factor of $2t$, yielding a clear asymptotic advantage as the number of conjunctive fields increases. For range queries, the computational and communication complexity of the *Index generation* phase in our scheme is multiplied by $(v + L_1)$ rather than by v alone, where L_1 denotes the number of elements in the encoding-1 set of the field over which the range query is performed, with L_1 bounded above by 8 for a one-digit decimal precision and by 11 for a two-digit decimal precision; in contrast, the Index-phase complexity of the other schemes remains unaffected by the presence of a range query. Similarly, for range queries, the computational and communication complexity of the *Trapdoor generation* phase, together with the computational complexity of the *Search/Test* phase, in our scheme are multiplied by $2L_0$, where L_0 denotes the number of elements in the encoding-0 set of the queried field and is bounded by the same maximum values as L_1 ; whereas for the other schemes, these quantities are instead multiplied by the total number of elements contained in the queried range, which grows substantially faster than L_0 or L_1 and thus imposes a significantly higher overhead as the range size increases.

6 Conclusions

This paper presents a blockchain-based framework for securing IoMT data, ensuring the preservation of confidentiality, searchability, integrity, availability, and fairness. Specifically, the proposed framework integrates a lightweight certificate-less asymmetric searchable encryption scheme, secure against IKGA, with a scalable public

Table 4: Comparison of different searchable encryption schemes in terms of computational complexity

Scheme	BuildIndex	Trapdoor	Search
[11]	$v(2H + 2P + E_{MG} + M_{MG})$	$2H + 2P + E_{MG} + M_{MG}$	$O(1)$
[13]	$v(4M_{AG} + 2A_{AG} + 2A_{zp} + 2H + 2E_{zp} + 2M_{zp})$	$4M_{AG} + 2H + 2A_{AG} + 3A_{zp} + 2M_{zp} + E_{zp} + 2A_{zp}$	$nv(5A_{AG} + 3M_{AG})$
[20]	$v(6H + 11M_{AG} + 5A_{AG} + M_{zp} + 2A_{zp})$	$2H + 2M_{zp} + A_{zp} + 3M_{AG} + 2A_{AG}$	$nv(2P)$
[21]	$v(2P + 5H + 2M_{zp} + 4M_{AG} + 3H + 3M_{zp} + 5M_{AG} + P + 2A_{AG})$	$A_{AG}2M_{zp} + 2M_{AG}$	$nv(2P + 2M_{zp} + M_{AG} + M_{MG})$
[36]	$v(2P + 7E_{MG} + 6M_{zp} + H + 3E_{MG})$	$7M_{zp} + E_{MG} + H + 2M_{zp} + E_{MG}$	$nv(2E_{MG} + P + M_{MG})$
[24]	$v(2M_{AG} + P + 3H + M_{zp} + M_{AG})$	$M_{AG} + M_{zp} + 2H + M_{AG}$	$nv(P + H)$
Ours	$v(2H + 2M_{AG} + 2M_{zp} + A_{zp} + A_{AG})$	$2H + 2M_{AG} + 2M_{zp} + A_{zp} + A_{AG}$	$O(1)$

n	Number of documents	E_{MG}	Exponentiation in multiplicative group
v	Number of distinct keywords	E_{zp}	Exponentiation in z_p^*
P	Symmetric bilinear pairing	A_{AG}	Addition in additive group
H	Hash operation	A_{zp}	Addition in z_p^*
M_{AG}	Multiplication in additive group	M_{zp}	Multiplication in z_p^*
M_{MG}	Multiplication in multiplicative group		

Table 5: Communication costs between different schemes

Scheme	Index generation	Trapdoor generation
[11]	$v.l$	l
[13]	$v.(2 G_{AG} + z_p)$	$2 G_{AG} + z_p $
[20]	$v.(4 G_{AG})$	$2 G_{AG} $
[21]	$v.(2 G_{MG} + 2 G_{AG} + G_{MG} + z_p)$	$3 G_{AG} + z_p $
[36]	$v.(3 G_{MG} + 3G_{MG})$	$ G_{MG} + 2 z_p + G_{MG} + 2 z_p $
[24]	$v.(G_{AG} + z_p + G_{AG} + z_p)$	$ G_{AG} + G_{AG} $
Ours	$v.l$	l

blockchain solution to ensure data integrity across multiple CSPs. Furthermore, formal security proofs are provided in the random oracle model, while SC are deployed to validate system performance. Experimental results indicate that the approach significantly reduces search latency and transactional overhead, scales effectively to large IoMT datasets, and maintains robust protection against key threats. Collectively, this work presents a practical, efficient, and comprehensive solution for next-generation smart healthcare systems, thereby reinforcing stakeholder trust and facilitating timely, data-driven clinical decision-making.

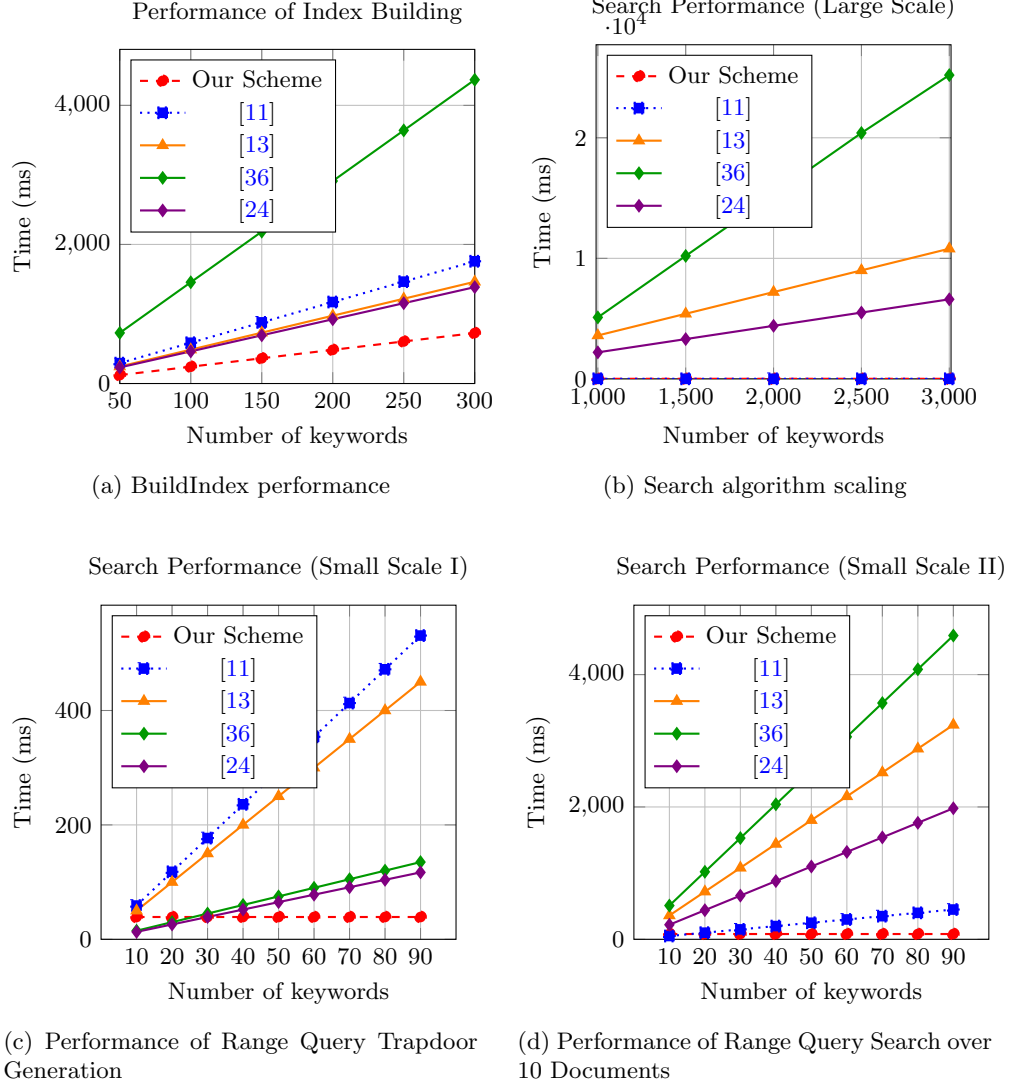


Fig. 2: Comparison of execution time across different algorithms.

The proposed scheme supports range queries through an implementation based on iterative coding and looping, which inherently constrains efficiency. There is a need for a more efficient and mathematically rigorous design, potentially utilizing techniques such as Order-Preserving Encryption (OPE), which allows the encryption of data so that the order of the plaintexts is preserved in the ciphertexts. Additionally, a comprehensive authentication and access control scheme is required, particularly one that aligns with established identity standards, such as the Electronic Identification, Authentication, and Trust Services (eIDAS). The current approach is tailored for

multiple writer and single reader scenarios; therefore, distinct schemes are necessary to address the more complex multiple writer and multiple reader environments effectively.

7 Declarations

7.1 Ethics Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

7.2 Conflict of Interest

Arian Arabnouri, as the first author, declares that he has no conflict of interest. Further, Alireza Shafeinejad, as the second author, declares that he has no conflict of interest.

7.3 Data Availability

This study is primarily based on formal verification and theoretical validation methods. As no empirical datasets were generated or analyzed during this research, this statement is not applicable.

7.4 Author Contribution

Arian Arabnouri wrote the main manuscript text and searchable encryption scheme and Alireza Shafeinejad validated the work and prepared the revised manuscript. All authors reviewed the manuscript.

7.5 Funding

Not applicable.

7.6 Consent to publish

All authors have approved the manuscript and agree with its submission to Peer-to-Peer Networking and Applications.

References

- [1] Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: International Conference on the Theory and Applications of Cryptographic Techniques, pp. 506–522 (2004). Springer
- [2] Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: International Conference on Computational Science and Its Applications, pp. 1249–1259 (2008). Springer

- [3] Yau, W.-C., Heng, S.-H., Goi, B.-M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: International Conference on Autonomic and Trusted Computing, pp. 100–105 (2008). Springer
- [4] Hu, C., Liu, P.: A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension. In: International Conference on Computer Science, Environment, Ecoinformatics, and Education, pp. 131–136 (2011). Springer
- [5] Yau, W.-C., Phan, R.C.-W., Heng, S.-H., Goi, B.-M.: Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. *International Journal of Computer Mathematics* **90**(12), 2581–2587 (2013)
- [6] Chen, Y.-C.: SPEKS: secure server-designation public key encryption with keyword search against keyword guessing attacks. *The Computer Journal* **58**(4), 922–933 (2015)
- [7] Shiraly, D., Eslami, Z., Pakniat, N.: Designated-tester identity-based authenticated encryption with keyword search with applications in cloud systems. *Journal of Systems Architecture* **152**, 103183 (2024)
- [8] Yanguo, P., Jiangtao, C., Changgen, P., Zuobin, Y.: Certificateless public key encryption with keyword search. *China Communications* **11**(11), 100–113 (2014)
- [9] Chenam, V.B., Ali, S.T.: A certificateless authenticated searchable encryption with dynamic multi-receiver for cloud storage. *Computer Communications* **211**, 157–177 (2023)
- [10] Chenam, V.B., Ali, S.T.: A pairing-free dynamic multi-receiver certificateless authenticated searchable encryption for cloud storage. *Journal of Information Security and Applications* **83**, 103801 (2024)
- [11] Arabnouri, A., Shafieinejad, A.: BACASE-SH: Blockchain-based authenticated certificate-less asymmetric searchable encryption for smart healthcare. *Peer-to-Peer Networking and Applications* **17**(4), 2298–2314 (2024)
- [12] Long, W., Zeng, J., Wu, Y., Gao, Y., Zhang, H.: A certificateless verifiable bilinear pair-free conjunctive keyword search encryption scheme for IoMT. *Electronics* **13**(8), 1449 (2024)
- [13] Liu, X., Sun, Y., Dong, H.: A pairing-free certificateless searchable public key encryption scheme for IoMT. *Journal of Systems architecture* **139**, 102885 (2023)
- [14] Cheng, L., Qin, J., Meng, F.: Privacy leakage of certificateless public key authenticated searchable encryption via frequency analysis: Attacks and revises. *Computer Standards & Interfaces* **87**, 103762 (2024)

- [15] Zhang, L., Zhou, F., Zhang, Q., Xiong, W., Tian, Y.: An improved secure designated server certificateless authenticated searchable encryption scheme for IIoT. *High-Confidence Computing* **5**(3), 100301 (2025)
- [16] Chenam, V.B., Ali, S.T.: A designated tester-based certificateless public key encryption with conjunctive keyword search for cloud-based MIIoT in dynamic multi-user environment. *Journal of Information Security and Applications* **72**, 103377 (2023)
- [17] Elhabob, R., Taha, M., Xiong, H., Khan, M.K., Kumari, S., Chaudhary, P.: Pairing-free certificateless public key encryption with equality test for internet of vehicles. *Computers and Electrical Engineering* **116**, 109140 (2024)
- [18] Tian, J., Lu, Y., Li, J.: Lightweight searchable and equality-testable certificateless authenticated encryption for encrypted cloud data. *IEEE Transactions on Mobile Computing* **23**(8), 8431–8446 (2024)
- [19] Tsai, T.-T., Yang, J.-H.: Leakage-resilient certificateless public key encryption with equality test resistant to side-channel attacks. *Computer Networks* **270**, 111485 (2025)
- [20] Li, H., He, D., Vijayakumar, P., Alqahtani, F., Tolba, A.: A certificateless and KGA-secure searchable encryption scheme with constant trapdoors in smart city. *Digital Communications and Networks* (2024)
- [21] Liu, X., Liu, P., Yang, B., Chen, Y.: One multi-receiver certificateless searchable public key encryption scheme for IoMT assisted by LLM. *Journal of Information Security and Applications* **90**, 104011 (2025)
- [22] Zhou, Y., Zhao, T., Jing, Z., Meng, W., Gu, C., Hu, H.: PCL-BPRE: Privacy-preserving certificateless-based broadcast proxy re-encryption for data sharing in cloud-based IIoT. *Future Generation Computer Systems*, 108307 (2025)
- [23] Guo, H., Li, X., Qin, H.: A lightweight and secure certificateless encryption scheme for smart city IoT security. *Discover Computing* **28**(1), 104 (2025)
- [24] Taha, M., Zhong, T., Elhabob, R., Xiong, H., Amoon, M., Kumari, S.: Certificateless searchable encryption with cryptographic reverse firewalls for IIoT. *Computer Standards & Interfaces* **95**, 104034 (2026)
- [25] Zou, Y., Peng, T., Wang, G., Luo, E., Xiong, J.: Blockchain-assisted multi-keyword fuzzy search encryption for secure data sharing. *Journal of Systems Architecture* **144**, 102984 (2023)
- [26] Tan, X., Cheng, W., Huang, H., Jing, T., Wang, H.: Edge-aided searchable data sharing scheme for IoV in the 5G environment. *Journal of Systems Architecture* **136**, 102834 (2023)

- [27] Yin, H., Zhao, Y., Zhang, L., Qiao, B., Chen, W., Wang, H.: Attribute-based searchable encryption with decentralized key management for healthcare data sharing. *Journal of Systems Architecture* **148**, 103081 (2024)
- [28] Li, H., Deng, L., Gu, Y., Wang, N., Chen, Y., Li, S.: A secure data sharing scheme based on searchable public key encryption for authorized multi-receiver. *Journal of Systems Architecture* **167**, 103489 (2025)
- [29] Han, Y., Han, J., Meng, W., Lai, J., Wu, G.: Blockchain-based privacy-preserving public key searchable encryption with strong traceability. *Journal of Systems Architecture* **155**, 103264 (2024)
- [30] Jin, C., Lu, W., Kong, L., Wang, J., Liu, X., Chen, G., Zhang, H., Weng, J.: Blockchain-based multi-user dynamic verifiable searchable encryption for secure data storage and query on malicious cloud server. *Journal of Systems Architecture*, 103647 (2025)
- [31] Jin, C., Li, J., Sun, Y., Ji, L., Liu, X., Chen, H., Zhou, L., Zhai, Z., Zhang, H., Weng, J.: DSSE-DMVS: A blockchain-based dynamic searchable symmetric encryption supporting multi-keyword queries and data deduplication. *Journal of Systems Architecture*, 103540 (2025)
- [32] Wang, J., Ma, S., Huang, Q.: Verifiable keyword searchable encryption with enhanced security using aggregate key in blockchain-based smart healthcare. *Journal of Information Security and Applications* **93**, 104173 (2025)
- [33] Ghosh, S., Islam, S.H., Vasilakos, A.V.: Blockchain-assisted provably secure identity-based public key encryption with keyword search scheme for medical data sharing. *Journal of Systems Architecture*, 103619 (2025)
- [34] Lin, H.-Y., Tzeng, W.-G.: An efficient solution to the millionaires' problem based on homomorphic encryption. In: *International Conference on Applied Cryptography and Network Security*, pp. 456–466 (2005). Springer
- [35] Merkle, R.C.: A digital signature based on a conventional encryption function. In: *Conference on the Theory and Application of Cryptographic Techniques*, pp. 369–378 (1987). Springer
- [36] Zhou, R., He, Y., Li, W., Wang, Y., Zhang, X.: Subversion-resistant public-key searchable encryption for data sharing in iiot. *Journal of Systems Architecture* **161**, 103370 (2025)

Algorithm 6 Verify Function

Require: $T_c, \mathcal{T}_{(c,v)}, \mathcal{PT}$ **Ensure:** 0 or an error code

```
1: sort  $\mathcal{PT}$  in ascending order by upload time
2:  $DC \leftarrow \mathcal{PT}.length$ 
3: for  $i \leftarrow 0$  to  $DC - 1$  do
4:   if  $\mathcal{PT}_i.ID \neq Hash(\mathcal{PT}_i.\xi)$  then
5:     return err.TAMPERED_DOC
6:   end if
7:    $\mathcal{H} \leftarrow \text{Invoke } readData(\mathcal{PT}_i.cr) \text{ on SC}$ 
8:    $\mathcal{H} \leftarrow \text{Invoke } readData(\mathcal{PT}_i.cr - 1) \text{ on SC}$ 
9:    $K \leftarrow \mathcal{H}.key$ 
10:   $\hat{K} \leftarrow \mathcal{H}.key$ 
11:  if  $\mathcal{H} \neq null$  and  $\hat{K} \neq hash(K)$  then
12:    return err.INVALID_KEY
13:  end if
14:   $id \leftarrow \mathcal{PT}_i.ID$ 
15:   $cond1 \leftarrow \mathcal{T}_{(c,v)} \neq null$ 
16:   $cond2 \leftarrow (i = 0)$ 
17:   $cond3 \leftarrow \mathcal{PT}_i.cr \neq \mathcal{PT}_{i+1}.cr$ 
18:   $cond4 \leftarrow C_2 \neq \mathcal{H}.U.U.C_{(c,v)}$ 
19:   $cond5 \leftarrow \Theta \neq \mathcal{H}.U.U.\theta_{(c,v)}$ 
20:   $cond6 \leftarrow \mathcal{PT}_i.cr \neq \mathcal{PT}_{i+1}.cr$ 
21:   $cond7 \leftarrow C_1 \neq \mathcal{H}.U.C_c$ 
22:   $cond8 \leftarrow \Delta \neq \mathcal{H}.U.\delta_c$ 
23:  if  $cond1$  and  $cond2$  then
24:     $C_2 \leftarrow \mathcal{H}.U.U.C_{(c,v)}$ 
25:     $\Theta \leftarrow hmac(K, id, \mathcal{H}.U.U.\theta_{(c,v)}, \mathcal{T}_{(c,v)}, C_2)$ 
26:  else if  $cond1$  and  $\neg cond2$  then
27:     $C_2 \leftarrow C_2 + 1$ 
28:     $\Theta \leftarrow hmac(K, id, \Theta, \mathcal{T}_{(c,v)}, C_2)$ 
29:  else if  $cond1$  and  $cond3$  and  $cond4$  then
30:    return err.INCOMPLETE
31:  else if  $cond1$  and  $cond3$  and  $cond5$  then
32:    return err.TAMPERED
33:  else if  $\neg cond1$  and  $cond2$  then
34:     $C_1 \leftarrow \mathcal{H}.U.C_c$ 
35:     $\Delta \leftarrow hmac(K, id, \mathcal{H}.U.\delta_c, T_c, C_1)$ 
36:  else if  $\neg cond1$  and  $\neg cond2$  then
37:     $C_1 \leftarrow C_1 + 1$ 
38:     $\Delta \leftarrow hmac(K, \mathcal{PT}_i.ID, \Delta, T_c, C_1)$ 
39:  else if  $\neg cond1$  and  $cond6$  and  $cond7$  then
40:    return err.INCOMPLETE
41:  else if  $\neg cond1$  and  $cond6$  and  $cond8$  then
42:    return err.TAMPERED
43:  end if
44: end for
45: return 0
```

Algorithm 7 Retrieval Function

Require: $gp, \mathcal{K}, pk_{DU}, sk_{DO}, (c_1, r_1), \dots, (c_n, r_n), s^T, e^T$

Ensure: $\mathcal{R}$

```
1:  $\mathcal{RS} \leftarrow DU\text{-Search}(gp, pk_{DU}, sk_{DO},$   
    $(c_1, r_1), \dots, (c_n, r_n), s^T, e^T)$   
2:  $\mathcal{R} \leftarrow \emptyset$   
3: for all  $(T_c, \mathcal{T}_{(c,v)}, \mathcal{D}) \in \mathcal{RS}$  do  
4:    $code \leftarrow Verify(T_c, \mathcal{T}_{(c,v)}, \mathcal{D})$   
5:   if  $code = 0$  then  
6:     for all  $doc \in \mathcal{D}$  do  
7:        $\mathcal{PT} \leftarrow Decrypt(\mathcal{K}, doc.\xi_{Doc_i})$   
8:        $\mathcal{R} \leftarrow \mathcal{R} \cup \{(T_c, \mathcal{T}_{(c,v)}, \mathcal{PT})\}$   
9:     end for  
10:  else  
11:     $\mathcal{R} \leftarrow \mathcal{R} \cup \{(T_c, \mathcal{T}_{(c,v)}, code)\}$   
12:  end if  
13: end for  
14: return  $\mathcal{R}$ 
```
