IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429

Volume 5, Issue 5, May 2025

Impact Factor: 7.67

Enhancing Data Warehouse Queries through Intelligent Keyword Search Techniques

Dr. U. Nilabarnisha[1], Chandru. S[2], Hariharan. S[3], Pugazhenthi. M[4], Kamesh. V[5]

Faculty, Computer Science and Engineering[1]

Students, Computer Science and Engineering[2,3,4,5]

Mahendra Institute of Engineering and Technology, Salem, India

Abstract: *Cloud data warehouse (CDW) platforms have been offered by many cloud service providers to provide abundant storage and unlimited accessibility service to business users. Sensitive data warehouse (DW) data consisting of dimension and fact data is typically encrypted before it is outsourced to the cloud. However, the query over encrypted DW is not practically supported by any analytical query tools. The Searchable Encryption (SE) technique is palpable for supporting the keyword searches over the encrypted data. Although many SE schemes have introduced their own unique searching methods based on indexing structure on top of searchable encryption techniques, there are no schemes that support Boolean expression queries essential for the search conditions over the DW schema. In this paper, we propose a secure and verifiable searchable encryption scheme with the support of Boolean expressions for CDW. The technical construct of the proposed scheme is based on the combination of Partial Homomorphic Encryption (PHE), B+Tree and Inverted Index, and bitmapping functions to enable privacy-preserving SE with efficient search performance suitable for encrypted DW. To enhance the scalability without requiring a third party to support the verification of search results, we employed blockchain and smart contracts to automate authentication, search index retention, and trapdoor generation. For the evaluation, we conducted comparative experiments to show that our scheme is more proficient and effective than related works.*

Keywords: Cloud data warehouse

I. INTRODUCTION

Typically, a data warehouse (DW) serves as the repository for a wide array of sensitive or strategic data, where the aggregated outcomes are derived from a multidimensional framework and feature significantly larger data volumes. The cloud data warehouse (CDW) represents a promising platform that offers high resource resilience and accessibility for businesses. Since the cloud is honest but curious, data encryption techniques are generally applied before outsourcing the approving it for publication was Nitin Gupta. in which keywords are extracted from a data cube, encrypted, data to the cloud. Since the data warehouse is constructed based on multidimensional model where multiple dimensions

The associate editor coordinating the review of this manuscript and To support analytical queries over encrypted DW, the user needs to make a normal query, while the cube result should be returned in an encrypted format. Then, authorized users with a key can decrypt and access the plain query result. However, this makes it impractical for multiple query results. Searchable encryption (SE) techniques are viable for supporting multiple queries in an efficient manner. SE is a method and uploaded to the cloud. Keywords are shared between data owners and data users in the secure channel. Once a search query is made, the search function will be performed by cloud to find a matching keyword from the data user's request with the ones stored on the cloud. Some studies [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29] have proposed a solution to support multiple keyword searches, such as multi-keyword rank searches and range searches with the search structure of a normal index, an inverted index, or a Tree index. These works allow users to input more keywords than the traditional ones, which speeds upand facts are materialized. One of the common DW models supported by many online analytical processing (OLAP) tools is cube-based or multidimensional OLAP(MOLAP). In MOLAP, DW consists of a number of data cubes, where each cube represents the pre-computed view of the dimension and fact data. the searching process. Most of the papers in [3], [4], [5], [6], [10], [11], [12], [13], [15], [16], [17], [22], [24], [26], and [27] introduced their optimized inverted index to support multiple keyword searches, where the index is listed and mapped to each keyword of the encrypted data.Nevertheless, existing SE schemes are not well applicable for supporting efficient search over encrypted DW for several reasons. First, since the cube is constructed based on multiple dimensions and fact data, the multiple keyword-based SE is not adequate for the search. The Boolean search connecting multiple keywords from multiple-dimension data binding with indexing is required. Second, existing SE schemes usually rely on a particular search structure for indexing and a collection of documents as the searching object, which are inefficient to apply to encrypted DW. This is because DW has complex data types for each dimension, and any indexing must be adaptable to the various data types within the warehouse. Finally, most SE solutions allow any users to perform searches over the outsourced data as long as they are legitimate users. However, DW is generally used for supporting decision-making and the search result over certain sets of encrypted cubes should be limited to the specific group of users who have the right to make a query. Therefore, the privacy-preserving SE and indexing structure must be tailored to satisfy this requirement.

Regarding the search strategies, tree-based indexing techniques such as B+Tree, Bitmap can handle more complex queries sucas fuzzy words and Boolean expressions. However, implementing such indexing techniques to support a large number of encrypted cubes together for a secure and verifiable search in a CDW setting is non-trivial. Various scenarios still present potential threats to search permission and the integrity of search results. For example, unauthorized individuals may attempt search queries, or search results could originate from unauthenticated sources or entities lacking proper permission. The privacy-preserving technique applied for indexing is therefore essential.

In this paper, we have introduced a secure and verifiable searchable encryption method with the support of Boolean expressions for encrypted data cubes outsourced in the cloud. Our proposed SE scheme is based on Partially Homomorphic Encryption (PHE) to ensure the security of keywords and three key indexing techniques, including B+Tree, inverted index, and bitmapping functions, along with. In addition, we applied blockchain technology to develop and execute smart contracts for enabling search permission and search result verification. The contributions of this article are summarized as follows:

1. We proposed a secure and fine-grained cryptographicbased access control scheme with efficient and verifiable searchable encryption for cloud data warehouse. Our proposed searchable encryption also supports Boolean expressions in the search query over encrypted data cubes outsourced in the cloud.
2. We introduced a novel design of indexing techniquesentailing the optimization of search space with the support of range and hierarchical search based on B+Tree indexing with the association of user role structure. In addition, we applied the inverted index and bitmapping to enable fast search for dynamic keyword searches and distinct values of the cube data, respectively.
3. We leveraged blockchain technology and smart contracts to support decentralized and robust user authentication, efficient indexing and search result verification of OLAP query, eliminating the need for third-party involvement in the verification process.
4. We conducted the comparative analysis and experiments to demonstrate the efficiency of our proposed scheme.

The remaining sections of this paper are organized as follows: Section II presents related works. Section III describes the background of materialized view, Paillier encryption, and blockchain. Section IV presents our proposed scheme. Section V describes our proposed cryptographic v construction. Section VI presents security analysis. Section VII discusses the evaluation and experiments. Section VIII concludes the paper.

II. RELATED WORK

There are several works that propose the technique of searchable encryption over encrypted data with the support of multiple keyword searches in various search structures and functionalities.

Typically, searchable encryption is based on two encryption approaches: symmetric and asymmetric encryption. For symmetric searchable encryption (SSE), symmetric encryption algorithm such as AES is used to encrypt and decrypt the search keyword. While SSE has been recognized for its efficiency and speed, the cost of key management is high if there are a large number of users. For asymmetric searchable encryption (ASE), the concept of key pairs is applied to the keyword in the way that a public key is used for encryption and a private key is used for decryption. Various forms of searchable encryption (ASE) have been examined in the underlying research area. For example, a public encryption with keyword search (PEKS), utilizes the public key to encrypt keywords extracted from data [1], [26]. Attribute Based Searchable Encryption (ABSE) [6], [13], [14], [15], [18], [29] involves the assignment of attributes to keyword indices. These attributes are then matched with user query trapdoors to maintain the confidentiality of keywords and the overall encryption characteristics. Additionally, Ciphertext Policy Attribute-Based Searchable Encryption (CP-ABE-SE) is a fine-grained and specialized method that adds an additional layer of security and facilitates complex multi-keyword searches in queries, as used in the scheme [23].

Recent works [5], [22], [27], [36] employed homomorphic encryption to support SE functions. Specifically, both full homomorphic encryption (FHE) [27] and partially homomorphic encryption (PHE) [5], [22], [36] have been adopted due to their ability to perform operations directly on encrypted data, eliminating the need for decryption. In the case where the basic search operations are needed and efficiency is a primary concern, PHE is a better choice.

In addition to the cryptographic method used as a core construct of SE, indexing search structures can be implemented to support efficient search. For instance, the inverted index is employed in schemes [3], [4], [5], [10], [12], [13], [15], [16], [17], [22], [24], [26], [27] which provide specific locations for the search within a dataset. When a user queries a term, the server promptly references the index, efficiently locating and retrieving the relevant documents. Basically, the B+Tree is regarded as the suitable indexing tree for hierarchical and range-based data types. It has been utilized in several schemes [9], [14], [25], [26], [28]. It supports fast queries and dynamic updates, insertion, and deletion, with encrypted indices being stored at leaf nodes as seen in the scheme [30], [31]. Another function to support the fast retrieval of indexing searches is bitmapping. It has been utilized in schemes [32], [33], [34] that are efficient for databases with limited distinct values. By transforming data into bit arrays, bitmap indexing can substantially reduce search costs.To provide more search capability, there are schemes that can support both multiple keyword and Boolean expressions [4], [21], [27] which deal with more complexity of the index structure and search conditions.

In [4], Zheng et al. introduced a system based on the obfuscating technique and dynamic symmetric searchable encryption that supports a single keyword with Boolean queries. The scheme retrieves bitmaps matching the queried keywords with the chosen anonymous parameter k. The client then computes the Boolean function on these bitmaps to determine the documents' identifiers that satisfy the Boolean query. In [6], the authors developed encrypted indexes for keyword sets associated with the stored data, which allow the cloud service provider (CSP) to perform searches on encrypted data withoutever accessing the plaintextkeywords, thereby ensuring data confidentially and privacy. Similarly, in [37], the authors did not mention the use of a standard search index, but they utilized cryptographic methods to ensure keyword searchability in the lightweight public key SE for mobile devices. In [21], the authors proposed the technique of three on-chain indexes: EDindex, BSindex, and PTindex. The ED index manages the storage of encrypted data with an inverted index. BSindex is used to support the calculation of stag and xtoken from the search query before they are compared with the index storing on the blockchain executed by smart contracts with PTindex. With this on-chain search procedure, smart contracts will check whether all x-tokens exist in the BSindex or not with the comparative formular.In[27],theauthorspresentedtheutilizationofTerm Frequency-Inverse Document Frequency (TF-IDF) for the purpose of arranging pertinent outcomes. They also incorporated techniques such as locality-sensitive hashing and bloom filters to facilitate a fuzzy keyword search, in addition to enhancing the bi-gramme keyword transformation approach. While this approach supports Boolean expressions, the accuracy of search results is lower than that of the systems that directly support Boolean expressions.

Recently, some SE works [8], [21], [35] integrated blockchain technology to offer robust search result verification as well as assist the user authentication process. Employing blockchain also provides transaction traceability and tamper resistance properties beneficial for maintaining trustworthy keyword indices for searchable encryption applications. In [8], Chen et al. proposed a verifiable searchable encryption approach that acquires verification components during trapdoor generation from user queries. This trapdoor is generated with authentication properties and is subsequently validated by the blockchain, serving as proof of the hashed keyword. The utilization of blockchain technology guarantees that search results remain unaltered. In [21], Wang et al. proposed the SE scheme designed to maintain the integrity of medical records. This is achieved through the execution of smart contracts, which also serve the dual role of managing access control for encrypted data by checking who can access and share it.In [35], Rong-Bing et al. proposed the utilization of blockchain technology for ensuring data integrity. This is done by creating an immutable ledger and managing searchable encryption indexes. This approach not only maintained the confidentiality and privacy of the data, but it also optimized search costs over large volumes of search queries and data sharing transactions.

Nonetheless, employing a single indexing technique directly to support searches across a large number of encrypted data cubes is not feasible. This is due to the high search space costs and the complexity of multidimensional data cubes. As a result, a comprehensive approach that combines Boolean multi-keyword searches, restricted user privilege search spaces, efficient range, and distinct search structures is promising but poses a real challenge.

III. PRELIMINARIES

This section describes the background of the materialized views concept which includes the definition of multidimensional space and base cube. Then, we briefly describe the Paillier encryption and blockchain technology.

A. MATERIALIZED VIEWS

In a data warehouse, materialized view (MV) is a pre-computed view result comprising aggregated and/or joined data from fact and possibly dimension tables. In MOLAP, a DW is modelled in a multidimensional space where multiple dimensions are formed and associated with the measure attribute. The precomputed view can be calculated from the possible aggregation operations of the dimensions and measured in a cube.

Definition 1:

Multidimensional space: Let be the space of all dimensions. For each dimension D,i there exists a set of levels, denoted as levels (Di). A dimension is a lattice (H, ≺) of levels. Each path in the lattice of a dimension hierarchy, beginning with its least upper bound, and ending with its greatest lower bound is called a dimension path. For example, the dimension path [day, week, month, year] is represented as day≺week≺ month≺ year.

Definition 2: Base Cube

A base cube Cbis a 3-tuple< D, L, R> where

• D= < D1, D2, ..., Dn, M> is a list of dimensions (Di, M ∈). M is a measure of the cube.

• L =<DL1,DL2,..., DLn, ∗ML> is a list of dimension levels (DLi., ∗ML ∈ 9). ML is the dimension level of the measure of the cube where the measure level (∗ML) belongs to a set 9. This set represents all possible measure levels within the data warehouse schema.

• R is a set of cell data formed as a tuple x = (x1, x2, ..., xn, ∗m) where I in [1, ..., n], xi ∈ dom(DLi) and ∗m ∈ dom(∗ML).

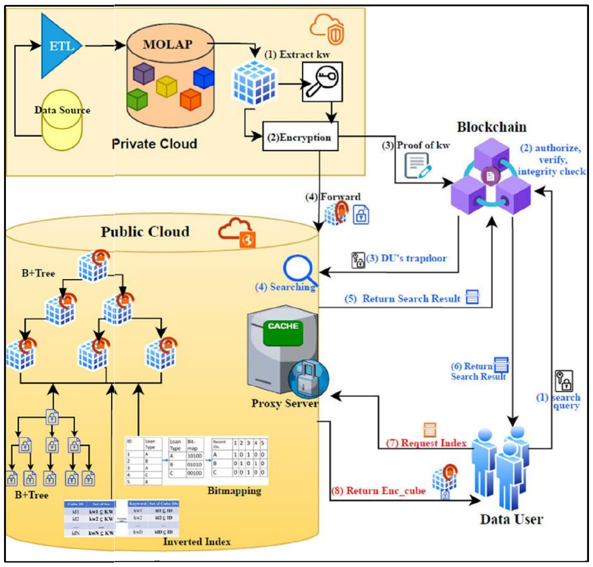
In our model, we assume that materialized view represents all possible views of the base cube Cb. Each view is computed from the set of aggregation operations including {sum, avg, count, max, min, rank(n)}. Each one of the operations results in a new cube c' or a materialized view (MV).

B. PAILLIER ENCRYPTION [36]

Paillier Encryption (PE) is the probabilistic asymmetric algorithm for public key cryptography. In PE, the message space M for the encryption is n. N is a product of two large prime numbers p and q.

Let L be defined as L(x) = (X − 1)/n. For a message m ∈ n, we denote [m] ∈ n2 to be the encryption of m with the public key pk. Particularly, Paillier encryption consists of three algorithms P ={P.KeyGen, P.Enc, P.Dec} which are defined as follows:

• P.KeyGen(1k): This algorithm is used to generate the public key. It begins by establishing an RSA modulus n = pq of k bits where p and q are large primes such that gcd(pq, (p-1)(q-1)) = 1. Let K = lcm((p-1) (q - 1)) = 1 and pick g ∈ ∗n. The public key is the pair pkp = (n, g) and the secret is skp = K.



Our system FIGURE 1. model.

• P.Encpkp(m):
• P.Decskp ([m]):
• This algorithm is employed to encrypt a messagem ∈ n : chooser ∈ and compute [m] = gm∗ rn mod n2 ∈n2.
• To decrypt a ciphertext c = [m], this algorithm computes m as follows: m = (L(msk)mod n2/ L(gsk) mod n2) mod n.

C. BLOCKCHAIN

Blockchain technology is an immutable, distributed, transparent, and traceable ledger that records the provenance of digital data. Its foundation lies in public key encryption and cryptographic hashing techniques. The digital assets or data stored within each block maintain their immutability due to the fact that once a block is finalized, it is hashed and interconnected with others in the blockchain network. In a typical blockchain structure, each block comprises essential elements, including a cryptographic hash of the preceding block, a timestamp indicating when the transaction took place, a nonce value, and the transaction data. On the blockchain, smart contracts, which are self- runnable programmes can be deployed and operated on a blockchain network.

IV. OUR PROPOSED SCHEME

In this section, we present the system model, our proposed indexing technique, and the construction of searchable encryption scheme.

A. SYSTEM MODEL

We proposed a secure and verifiable searchable encryption for cloud data warehouse. Figure 1 illustrates the system overview of our proposed scheme.

The system model consists of the following entities.

1. The Private Cloud Service Provider is responsible for storing the data cube, which is organized using MOLAP methodology following the ETL process, where data is extracted from various sources, transformed, and loaded. The data owners extract keywords from each data cube (MV) before subjecting them to encryption via a Paillier

cryptographic algorithm. Subsequently, all the encrypted data cubes (Enc_MV) are transmitted to the proxy server hosted in the public cloud.

2. Proxy Server is a semi-trusted server located in the cloud responsible for executing searches and returning search result indices to the blockchain. Additionally, itmaintainsamemorycacheforfrequentlyquerieddata within a specific timestamp to expedite search retrieval.

3. The Public Cloud Service Provider (Pub_CSP) is responsible for housing all the components related to Enc_MV, which is organized in a B+Tree structure to facilitate rapid searches. Enc_kw, the encrypted keywords, serves a triple-purpose function: 1) It extends the leaf nodes of the B+Tree as the parent tree to enable range and hierarchical searches. 2) It functions as a database or table for creating an inverted index for specific keywords. 3) It is used as a large table for bitmap indexing of distinct keyword values.

4. Blockchain platform serves as the repository for accessing and searching transaction records. It incorporates smart contracts that fulfill various roles, including storing evidence of keywords, validating user permissions, authorizing search queries to locate the index of Enc_MV related to the keyword and user's trapdoor, and conducting integrity checks.

5. Data Users (DUs) perform an OLAP query or search the keywords to get a particular Enc_MV.

B. OUR PROPOSED B+ TREE, INVERTED INDEX, AND BITMAP INDEXING FOR ENCRYPTED CUBES

Our proposed SE method comprises three combinations of indexing and search structures: B+tree, inverted index, and bitmap index. Each of these structures is designed to handle distinct types of data values associated with individual dimensions and factual data within the cube. To better grasp the concept of the data cube, Table 1 provides an example from a bank loan scenario, demonstrating the construction of multidimensional data.

In the context of the multidimensional data cube, as illustrated in Table 1 above, we construct all data cubes using the B+Tree data structure. In our design, there are 38,000 generated records for all data cubes, and this B+Tree structure greatly facilitates rapid retrieval, insertion, and deletion of data. In our design, the structure is associated with user privileges, where users can only query the cube that aligns with their role within the system. However, within each data cube, there can be thousands of records. The implementation of B+Tree search significantly narrows down the search space, leading to reduced time consumption when searching for specific records within a data cube. A sample B+Tree search structure is depicted in Figure 2 below.

TABLE 1. Example of A bank loan data cube.

No	Date	First Name	Last Name	Branch	Loan Type	Amount
1	2020-12 21	James	Smith	01	A	12000
2	2021-03 08	Mary	Johnson	01	B	23456
3	2022-07 19	John	Willaims	02	C	78910
4	2022-12 12	Jennifer	Brown	04	C	50000
5	2020-07 08	Michael	Jone	04	B	23500
6	2020-03 14	Robert	Davis	03	A	46800
7	2021-11 28	Linda	Miler	03	C	80000
8	2021-05 21	Jennifer	Garcia	02	C	91230
9	2022-09 11	William	Wilson	01	B	14300
10	2021-10 10	Smith	Taylor	03	A	77700

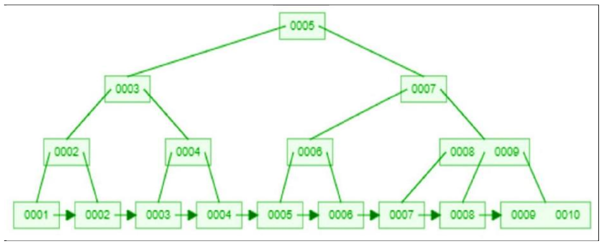
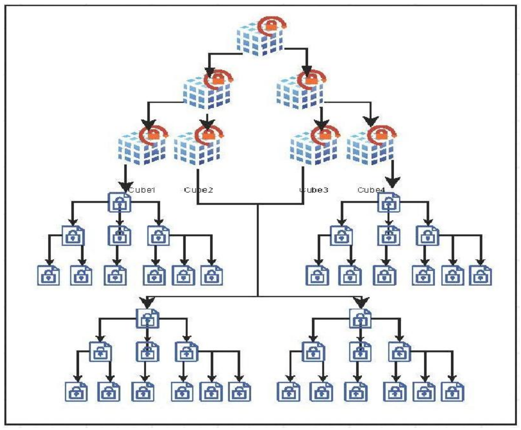
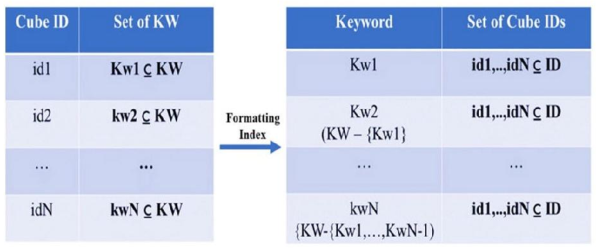


FIGURE 2. A sample of B+ tree structure.

The B+Tree depicted above has a maximum degree of 3, and each leaf node corresponds to a unique number of values in ascending order, connected by linked pointers. Each leaf node possesses a distinct node key number, which is assigned in ascending order from the smallest to the largest. A parent node may share the same unique node key value with one of its leaf nodes, yet this value essentially serves as an index number delineating the range of its child nodes. For example, a parent node with a node key value of 0006 may have child nodes with key values of 0005 and 0006. It is important to note that each child node maintains a unique node key value, ensuring a clear and orderly structure within the system. For instance, when a user queries for an amount ‘‘x’’ where x is less than 8 or greater than 2, the result would be returned from all leaf nodes where its node key value is range from 3 to7. Additionally, we integrate three indexing search functions for each data cube to efficiently retrieve data. These functions include the B+Tree, which facilitates range or hierarchical searches, similar to the parent B+Tree used for searching within a specific cube. The inverted index is employed for keyword-oriented attributes such as name or campus, and the bitmapping function supports searches for distinct values. Figure 3 illustrates the sub-B+Tree, which is one of the three combined search functions, serving as a subset of each leaf node of the main B+Tree.

In the initial setup, the parent B+Tree stores an encrypted data cube at each leaf node, and our proposed three indexing search functions are integrated for each cube. Consequently, when a user submits a query to retrieve records from any data cube, the query is divided into various search functions





Cube ID	Set of KW
id1	Kw1 ⊆ KW
id2	kw2 ⊆ KW
…	…
idN	kwN ⊆ KW

Formatting Index

Keyword	Set of Cube IDs
Kw1	id1,..,idN ⊆ ID
Kw2 (KW – {Kw1}	id1,..,idN ⊆ ID
…	…
kwN {KW-{Kw1,…,KwN-1)	id1,..,idN ⊆ ID

FIGURE 4. Example of inverted index.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 5, May 2025 Impact Factor: 7.67

that are embedded at each leaf node of the parent B+Tree structure. Additionally, we have another search function in the form of the inverted index, which is illustrated in Figure 4 below.

The inverted index proves valuable for attributes with a focus on keywords. From Figure 4, before we constructed the indexing format, we arranged the set of keywords (Set of KW) associated with 1 ID (Cube ID) per record in a row of an inverted index table. The Keywords (KwN) of each record can also be duplicated for a number of records themselves. Then, we formatted the index of each specific keyword (Keyword) associated with a list (set of Cube IDs) where a particular keyword is found in all IDs. For instance, if we have five records for customer names represented as {ID, LastName, FirstName} with values {[1, 'Mary', 'Johnson'], [2, 'Jennifer',

'Mary'], [3, 'Linda', 'Jennifer'], [4, 'Taylor', 'Mary'], [5, 'Linda', 'Johnson']}, we structure them as follows:

'Mary': {''Mary'': [{1}, {2}, {4}]},

'Johnson': {''Johnson'': [{1}, {5}]},

'Jennifer': {''Jennifer'': [{2}, {3}]},

'Linda': {''Linda'': [{3}, {5}]},

'Taylor': {''Taylor'': [{4}]}

The inverted index structure enables the grouping of multiple IDs into an index, with a dictionary storing those IDs that share the same string value, regardless of whether it pertains to LastName or FirstName. When a user queries for 'Mary' and 'Johnson', we point to the dictionary index of

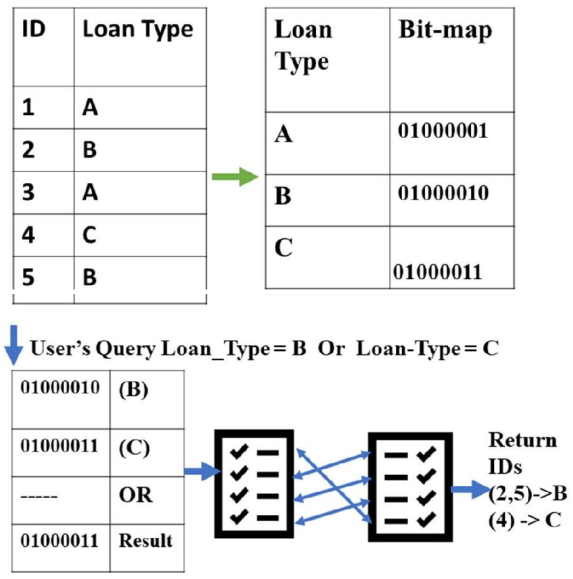


FIGURE 5. Bitmapping function.

{''Mary'': [{1}, {2}, {4}]} and {''Johnson'': [{1}, {5}]}, and the result is {''Mary AND Johnson'': [{1}]} representing the intersection based on the 'AND' operation.

To accommodate limited distinct values with Boolean operations, we introduce a bitmapping function that also supports Boolean expression searches. Figure 5 provides an example of how the bitmapping function operates.

The binary bitmapping function allows for highly efficient searches of any distinct value. As illustrated in Figure 5, the result from a user's query can be quickly identified by mapping the bit result to the structured documents. For example, iftheinputisLoan_TypeBorC,thebitmapvalueofeachloan will undergo an OR operation, producing a binary outcome. This outcome will then be assigned to the index location of the document according to its ID.

From the above three index searching structures, our proposed system can facilitate the search queries quickly and effectively because we handle the data types of each record efficiently, regardless of the query complexity.The user query will be broken down into 3 phases/functions, starting with B+Tree to handle the range and hierarchical data, inverted index for the value of attributes, and bitmapping for distinct values. The system returns the intersection of the output from those search functions as the final output.

C. SECURITY MODEL

Inthissection,wepresentthesecuritymodelforourproposed scheme. The security model defines the nature of the adversary, their capabilities, and the interactions between the data owner, authorized users, and the adversary within the proposed scheme. This security model is established according to the following adversarial model.

- Adversary Set: $A \subseteq Aall$ (A is a subset of all possible adversaries Aall).
- Adversary Type: A is a computationally bounded, passive adversary.
- C omputational Bound: The computational capabilities of Adversary A are bound in a manner preventing them from solving problems that necessitate both polynomial space and computational resources
- Active Attacks: A ∩ Active Attacks = 0 (A is limited to passive attacks and cannot engage in active attacks).

1) SEARCH QUERY MODEL

Adversary's Capabilities: Adversary A can submit search queries to the encrypted data and receive corresponding search results without learning the underlying data. A can also submit data to the encrypted index.

System Components:

1. Data Owner (DO)

- The data owner encrypts and stores the data using the Paillier encryption scheme.
- The data owner builds an index for efficient search and provides authorized users with search capabilities. For a given keyword and index I:
- DO →(Encrypt)keywordcipher = Paillier(keyword)
- DO →(Index)I(keyword)

2. Authorized Data Users (DUs) have the capability toperform searches on the encrypted data and retrieve relevant results without revealing the plaintext data. These users have a secret key for decryption.

- DU →(Search) Results (keywordcipher, q)
- DU→(decrypt) keywordplain =Paillier −1(keywordcipher)

Security Properties Confidentiality:

- The searchable encryption scheme guarantees the confidentiality of the data.
- A passive adversary should not be able to learn any information about the plaintext data from the encrypted data, index, or search queries.
- Formalized: A plaintext Search Privacy: An adversary should not be able to determine which terms are being searched.
- Formalized: A Info(queries) Index Privacy:
- The searchable index should not leak information about the data or the search terms, even when search queries are made.
- Formalized: A Info(index) Keyword Privacy:
- The scheme ensures the privacy of keywords used in search queries.
- Even if an adversary observes multiple search queries with overlapping keywords, they should not be able to deduce sensitive information about the data.

TABLE 2. Notation.

Notation	Description
K	A dictionary of every keyword
E(K)	A dictionary of every encrypted keyword
CombinedResult	The intersection of result from all search functions
p,q	Random large prime number
lcm	The least common multiple of (two random number) to calculate the lambda

V. OUR CRYPTOGRAPHIC CONSTRUCTION

The section presents the details and analyses of the DWMBSE construction. To ease of explanation, we define the notations used in our model as shown in Table 2 below.

Our scheme consists of ten major phases: system setup, keyword extraction, keyword encryption, data and keyword structure, user query process, trapdoor generation, search mechanism, blockchain result verification, user decryption, and data caching.

A. PHASE1: SYSTEM SETUP

In this phase, various components are set up, including the generation of public and private keys, a unique user ID for data user identification, a proof of keyword to be stored on the blockchain, and the configuration of cache memory on the proxy server located in the public cloud. While all cryptographic keys are generated by the Trusted Authority (TA), the remaining tasks are executed by the private cloud, withtheexceptionofcaching,whichismanagedbythepublic cloud. The system setup details are provided in Algorithm 1 as the following pseudo code:

Once the Algorithm 1 is executed, the following system components are created:

• Public Key and Private Key for Paillier Cryptography: The public and private keys required for Paillier cryptography are generated and ready for use in the system.

• Empty Dictionary for Proof of Keywords: An empty dictionary is set up to store proof of keywords. This dictionary will be used to securely store keywords on the blockchain.

• Unique ID for Each Data User: A unique identification (ID) is created for each data user. This ID will help identify and distinguish individual users within the system.

Algorithm 1 System Setup

```
1: systemSetup(()→public_key, private_key,
2: userDatabase, proofOfKeyword, cache){
3: # Choose two large prime numbers randomly
4: p,q ← while gcd (pq, (p-1)(q-1)) =1
5: n ← p × q
6: λ ← lcm(p-1, q-1)
7: g ← Random integer in Zn2
```

ISSN: 2581-9429 Volume 5, Issue 5, May 2025 Impact Factor: 7.67

```
 8: $\mu \leftarrow (L(g\lambda \text{ mod} n2))-1\text{mod} n$
9: return public key (n,q), private key($\lambda$,$\mu$)
10: public_key, private_key ← Paillier_setup()
11: userDatabase ← {}
 12: for each user do{
   13: userDatabase[user.ID] ←
   14: {''role'': user.role,
   15: ''public_key'': user.public_key}
 16: end for
 17: proofOfKeyword ← {}
 18: cache ← {}
 19: } end
Algorithm 2 Extract Keywords
1: Extract_keywords(records → K){
 2: K← {}
3: K.append(records(date[day, month, year]))
   4:K.append(records(customer[name, branch,
 5: loan_type]))
 6: K.append(records(amount[day, month, year]))
7: } end
```

• Empty Dictionary for Storing Search Result Index: Another empty dictionary is prepared to store the index of search results. This will be utilized in the memory cache on the proxy server to enhance search efficiency.
These components are fundamental to the system's operation, enabling secure keyword storage, user identification, and efficient search result retrieval.

B. PHASE2: KEYWORD EXTRACTION

In this stage, keywords are extracted from each data cube done in the private cloud. The keywords are divided based on their value type, representing each dimension of the multi-dimensional data cube stored in the data warehouse. The process is detailed in the following pseudo code:

c. PHASE3: KEYWORD ENCRYPTION AND FORWARDING

In this phase, the data owner applied Paillier encryption to the extracted keywords. The set of keywords, along with their associated Enc_kw and Enc_MV, is then distributed to various components: the proof of keyword is forwarded to the blockchain, and the encrypted keyword (Enc_kw) and encrypted data cube (Enc_MV) are sent to the proxy server in the public cloud. The detailed algorithm is presented in Algorithm 3 as follows:

```
Algorithm 3 Encrypt and Keywords Forwarding
1: encrypt_and_send_keywords((K, public_key)
2: →Encrypted_keywords, Proofs){
 3: E(K) ←{}
4: for each keyword in K do
 5: encryptedKeyword ←
    6:Paillier_Encrypted(keyword, public_key)
    7: E(K) )[keyword] ←encryptedkeyword
8: proofOfKeyword[keyword] ←
    9:Hash-SHA256(keyword)
 10: end for
 11: send_to_cloud(E(K))
 12: send_to_Blockchain(proofOfKeyword)
   13: } end
```

Algorithm 4 Strcture Encrypted Keywored

```
1: structure_keyword((E(K))→Inverted_index,
 2: Bitmap_index, B+Tree) {
3: Inverted_index ←create_Inverted_index(E(K))
4: Bitmap_index ←create_Bitmap_index(E(K))
5: B+Tree←create_B+Tree(E(K))
6: } end
```

D. PHASE5: USER QUERY PROCESS

After the system is fully set up, data users are able to submit search queries to the blockchain. The blockchain will either

Algorithm 5 Process User Query

```
1: process_User_Query((userID, query) →
2: (encrypted_search_results or error_message)){
 3: if NOT user_Identity_Check(userID,
 4: userDatabase) then
 5: return ‘‘Unauthorized User’’
6: end if
7: if query IS_EMPTY then
 8: return ‘‘Empty Query’’
9: end if
 10: trapdoor ← generate_Trapdoor (query,
   11:public_key)
 12: result ← search_and_verify (trapdoor)
 13: if result IS_NOT_Verified then
   14: return ‘‘Verification Failed’’
```

VI. EVALUATION

To evaluate our proposed scheme, we performed the comparative analysis by comparing the functional features and the computation cost of our scheme and three related works supportingsearchableencryptionincloud.Inaddition,wedid

TABLE 3. Functionality comparison.

Scheme	F1	F2	F3	F4	F5
4	✓	✓	X	X	X
8	✓	X	✓	✓	X

the experiments to demonstrate the search performance of our scheme and related works.

IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429 Volume 5, Issue 5, May 2025 Impact Factor: 7.67

A. FUNCTIONALITY COMPARISON

This section presents a comparison of the features of our proposed system and related works including [4], [8], and [30]. Table 3 presents a comparison between our scheme and these related works across five distinct functions.

As presented in Table 3, all schemes implemented lightweight encryption for the extracted keywords. For example, scheme [8], [30] utilized symmetric encryption while scheme [4] and ours relied on partial homomorphic encryption. For the scope of search operations, only scheme [4] and ours support multiple keyword searches and Boolean expressions, while scheme [8] and [30] do not support Boolean expressions. Additionally, it's important to note that only the BPVSE scheme [8] and our system utilize blockchain technology to enhance the authentication and verification processes for both data users and search results. Lastly, our scheme uniquely supports proxy search caching, a critical feature for rapidly retrieving search results, particularly when there's a high volume of identical and frequently requested queries. This feature significantly improves search performance, especially when dealing with large volumes of cube data that are frequently accessed.

B. COMPUTATION COST COMPARISON

This section compares computational cost between our work, scheme [4], [8], and [30] as presented in Table 4. To evaluate the cost for computing each property of each scheme, the following notations are used.

- |A0|: The number of attributes owned by the data owner.
- |AU|: The number of attributes owned by the data user.
- G0: exponentiation and XOR operations in group G0.
  G1: exponentiation in an elliptic curve group.
- Zp: the group {0, 1, ..., p-1} with multiplication modulo p.
- L: the number of iterations in searching for inverted index or/and bitmap index.
- B: the logarithm concerning the number of entries in the B+Tree.
- Esym: Represents the cost of symmetric encryption.
- |W|: the average number of keywords per document.
- |Q|: the number of keywords in the user's query.

TABLE 4. Computation cost comparison.

Scheme	Encryption	Trapdoor	Search	Decryption
[4]	$\|A_0\|+\|A_0\|G_0$	$\|A_U\|+\|A_U\|G_0$	$\|A_U\|B$	$\|A_U\|G_0$
[8]	$E_{sym}(Z_p)$	$\|A_U\|(2Z_p+G_0)$	G_0	$E_{sym}(Z_p)$
[30]	$E_{sym}\|A_0W\|$	$\|Q\|(E_{sym}+Z_p)$	$B+\|A_0\|\|W\|Z_p$	$E_{sym}\|A_0W\|$
Ours	$\|A_0\|(2Z_p+G_0)$	$\|A_U\|(Z_p+G_0)$	$\|A_U\|(L+B)$	$\|A_U\|(G_0+2Z_p)$

Scheme [4] and our scheme share similar computational costs, with encryption and associated expenses generally dependent on the number of attributes and exponentiation in G0 while schemes [8] and [30] deal with the cost of symmetric encryption and decryption. Specifically, scheme [4] additionally uses multiple XOR operations that correspond to the number of keywords or attributes. In contrast, our scheme incorporates partially homomorphic encryption (PHE), which is considered lightweight compared to fully homomorphic encryption (FHE). The cost of generating a trapdoor does not significantly differ across all schemes. Given the similar encryption costs, there is a slightly higher computational cost for the trapdoor generation in scheme [8], where additional verification processes, are involved in generating the trapdoor. In terms of search costs, only scheme [8] does not support multiple keywords and Boolean expressions, making it cost-efficient when dealing with single keywords. On the other hand, in scheme [30], the search cost is higher compared to scheme [4] and our scheme, particularly when handling a larger number of keywords per document, involving several multiplications in Zp. With regard to search structures, all schemes, except for scheme [8], implement a B+Tree index search structure to support multiple keyword searches. However, only scheme [4] and our scheme offer support for both multiple keywords and Boolean expressions, incurring comparable computational costs. Our scheme has a slightly higher cost than the scheme [4] due to the integration of three different indexing search functions. Our scheme is slightly higher than scheme [4] due to our combination of three different indexing search functions.

C. EXPERIMENTAL EVALUATION

In this section, we conducted experiments to measure the processing time for data cube generation, encryption, decryption, trapdoor generation, search, and query throughput. In addition, we measured the gas used in executing the smart contracts.

The implementation is done via Python's Cryptography and its standard libraries modules such as random, hashlib, csv, os, time, concurrent.futures, multiprocessing, pickle, threading, and datetime. Additionally, we employed third-party libraries such as phe [38] for the Paillier cryptographic system, web3 [39] for binding Python language with

TABLE 5. Time cost of major operations.

Operation	T_C	T_D	T_E	T_V
Time (ms)	202.9170	0.01072	0.002861	0.000476

TABLE 6. Processing time computation.

Functions	Enc.	Trapdoor	Verification	Dec.
Time (ms)	$T_C + n(T_E)$	$T_D + n(T_E)$	$n(T_V)$	T_C

Ethereum. We also used machine learning in Python called Scikit-learn [40] to stimulate the scheme [30]. The experiments were done on an Intel(R) Xeon(R) E-2336 CPU @ 2.9GHz and 16 GB of RAM on a server that is running on the Ubuntu 20.04 Operating System. We employed the Ethereum network as the blockchain platform for our simulation and utilized Solidity to develop the smart contracts. The development was carried out on Remix, which is a web-based Integrated Development Environment (IDE) designed for the Ethereum network. We utilized Ethereum's smart contracts as it fully utilizes the implementation of decentralized access control and transparent auditable operations mechanisms. This could allow for fine-grained control over who can access, modify, or query the data stored in the cloud data warehouse, reducing reliance on centralized entities for access management.

Performance Analysis

We first did the experiment to measure the cost of major operations, including encryption and decryption time (TC), trapdoor gen time (TD), and verification time (TV) of our proposed scheme. Table 5 shows the time used for running these operations. Table 6 presents how the time cost for each operation is computed.

In this paper, we conducted simulations of our proposed system to calculate the time required to perform the core functions of our system, such as keyword and search result encryption, trapdoor generation, verification, and search result decryption. As demonstrated in Table 5 , TC represents the time cost of using Paillier encryption and decryption for keywords and search results, which consistently takes around 202 milliseconds. For encryption and trapdoor generation, the timecostincreaseswiththenumberofrecordsnandarandom value TE. In our scheme, TC only is the time taken to perform decryption, while TD is the time taken to generate a trapdoor when the user makes a search query. Lastly, the TV is the time needed to verify the search result based on a hash- proof comparison.

Search performance

We did the experiment to compare the search performance of our scheme, [4], [8], and [30]. For the test, we varied the number of records contained in the data cube and measured the time used to complete the search process. In our experiment, we used Tiny OLAP open-source GitHub [40] to generate the 38,000 records for all data cubes.

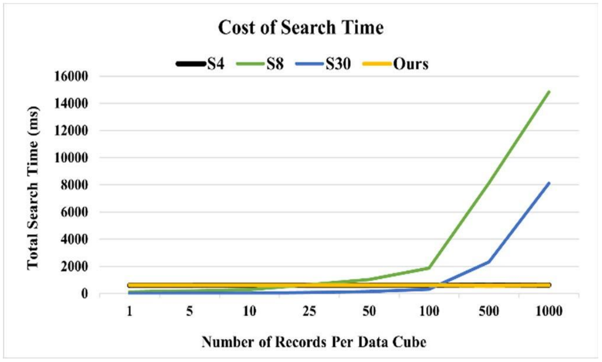


FIGURE 6. Cost of search time comparison.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a flexible, verifiable, and secure searchable encryption scheme with support for boolean expression over encrypted data cubes within a cloud-based data warehouse. Our scheme enjoys both security and search performance based on the integration of partial homomorphic encryption, inverted index, and B+Tree. In addition, we leveraged blockchain technology to streamline the automation of search permission verification, user authentication, and search result validation processes. These tasks are executed in a manner that ensures scalability and immutability. Notably, we have utilized various search function types to suit different data types applicable for searching over multidimentional data, such as inverted indexes, B+Trees, and bitmapping functions. Another key advantage of our proposed B+Tree indexing scheme is to reduce the search space. Our experiments have demonstrated that our scheme can significantly save time and resources. The system can also provide reasonable system throughput for supporting multiple concurrent OLAP query requests. For future works, we will investigate the technique to achieve fully forward security in supporting the keyword update.

REFERENCES

[1] H. Yin, W. Zhang, H. Deng, Z. Qin, and K. Li, ''An attributebased searchable encryption scheme for cloud-assisted IIoT,'' IEEE Internet Things J., vol. 10, no. 12, pp. 11014–11023, Jun. 2023, doi: 10.1109/JIOT.2023.3242964.

[2] X. Liu, H. Dong, N. Kumari, and J. Kar, ''A pairing-free certificateless searchable public key encryption scheme for industrial Internet of Things,'' IEEE Access, vol. 11, pp. 58754–58764, 2023, doi: 10.1109/ACCESS.2023.3285114.

[3] S. Guo, H. Geng, L. Su, S. He, and X. Zhang, ''A rankable Boolean searchable encryption scheme supporting dynamic updates in a cloud environment,'' IEEE Access, vol. 11, pp. 63475–63486, 2023, doi: 10.1109/ACCESS.2023.3284904.

[4] Y. Zheng, R. Lu, J. Shao, F. Yin, and H. Zhu, ''Achieving practical symmetric searchable encryption with search pattern privacy over cloud,'' IEEE Trans. Services Comput., vol. 15, no. 3, pp. 1358–1370, May 2022, doi: 10.1109/TSC.2020.2992303.

[5] Y. Wang, S.-F. Sun, J. Wang, J. K. Liu, and X. Chen, ''Achieving searchable encryption scheme with search pattern hidden,'' IEEE Trans. Services Comput., vol. 15, no. 2, pp. 1012–1025, Mar. 2022, doi: 10.1109/TSC.2020.2973139.

J. Li, X. Lin, Y. Zhang, and J. Han, ''KSF-OABE: Outsourced attributebased encryption with keyword search function for cloud storage,''

[6] IEEE Trans. Services Comput., vol. 10, no. 5, pp. 715–725, Sep. 2017, doi: 10.1109/TSC.2016.2542813.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 5, Issue 5, May 2025 Impact Factor: 7.67

[7] Q. Zhang, S. Wang, D. Zhang, J. Sun, and Y. Zhang, ‘‘Authorized data secure access scheme with specified time and relevance ranked keyword search for industrial cloud platforms,’’ IEEE Syst. J., vol. 16, no. 2, pp. 2879–2890, Jun. 2022, doi: 10.1109/JSYST.2021. 3093623.
[8] B. Chen, T. Xiang, D. He, H. Li, and K. R. Choo, ‘‘BPVSE: Publicly verifiable searchable encryption for cloud-assisted electronic health records,’’ IEEE Trans. Inf. Forensics Security, vol. 18, pp. 3171–3184, 2023, doi: 10.1109/TIFS.2023.3275750.
[9] J. Fu, N. Wang, B. Cui, and B. K. Bhargava, ‘‘A practical framework for secure document retrieval in encrypted cloud file systems,’’ IEEE Trans. Parallel Distrib. Syst., vol. 33, no. 5, pp. 1246–1261, May 2022, doi: 10.1109/TPDS.2021.3107752.
[10] L. Chen, Y. Xue, Y. Mu, L. Zeng, F. Rezaeibagha, and R. H. Deng, ‘‘CASE-SSE: Context-aware semantically extensible searchable symmetric encryption for encrypted cloud data,’’ IEEE Trans. Services Comput., vol. 16, no. 2, pp. 1011–1022, Mar. 2023, doi: 10.1109/TSC.2022. 3162266.
[11] R. Zhou, X. Zhang, X. Wang, G. Yang, H.-N. Dai, and M. Liu, ‘‘Deviceoriented keyword-searchable encryption scheme for cloud-assisted industrial IoT,’’ IEEE Internet Things J., vol. 9, no. 18, pp. 17098–17109, Sep. 2022, doi: 10.1109/JIOT.2021.3124807.
[12] L. Xue, ‘‘DSAS: A secure data sharing and authorized searchable framework for e-Healthcare system,’’ IEEE Access, vol. 10, pp. 30779–30791, 2022, doi: 10.1109/ACCESS.2022.3153120.
[13] Y. Yang, R. H. Deng, W. Guo, H. Cheng, X. Luo, X. Zheng, and C. Rong, ‘ ‘Dualtraceabledistributedattribute-basedsearchableencryptionandownership transfer,’’ IEEE Trans. Cloud Comput., vol. 11, no. 1, pp. 247–262, Jan. 2023, doi: 10.1109/TCC.2021.3090519.
[14] P. Zhang, Y. Chui, H. Liu, Z. Yang, D. Wu, and R. Wang, ‘‘Efficient and privacy-preserving search over edge–cloud collaborative entity in IoT,’’ IEEE Internet Things J., vol. 10, no. 4, pp. 3192–3205, Feb. 2023, doi: 10.1109/JIOT.2021.3132910.
[15] J. Liu, Y. Li, R. Sun, Q. Pei, N. Zhang, M. Dong, and V. C. M. Leung, ‘‘EMK-ABSE: Efficient multikeyword attribute-based searchable encryption scheme through cloud-edge coordination,’’ IEEE Internet Things J., vol. 9, no. 19, pp. 18650–18662, Oct. 2022, doi: 10.1109/JIOT.2022.3163340.
[16] Q. Liu, Y. Tian, J. Wu, T. Peng, and G. Wang, ‘‘Enabling verifiable and dynamic ranked search over outsourced data,’’ IEEE Trans. Services Comput., vol. 15, no. 1, pp. 69–82, Jan. 2022, doi: 10.1109/TSC.2019.2922177.
[17] G. Liu, G. Yang, S. Bai, H. Wang, and Y. Xiang, ‘‘FASE: A fast and accurate privacy-preserving multi-keyword top-k retrieval scheme over encrypted cloud data,’’ IEEE Trans. Services Comput., vol. 15, no. 4, pp. 1855–1867, Jul. 2022, doi: 10.1109/TSC.2020.3023393.
[18] M. Zeng, H. Qian, J. Chen, and K. Zhang, ‘‘Forward secure public key encryption with keyword search for outsourced cloud storage,’’ IEEE Trans. Cloud Comput., vol. 10, no. 1, pp. 426–438, Jan. 2022, doi: 10.1109/TCC.2019.2944367.
[19] Z.-Y. Liu, Y.-F. Tseng, R. Tso, Y.-C. Chen, and M. Mambo, ‘‘Identitycertifying authority-aided identity-based searchable encryption framework in cloud systems,’’ IEEE Syst. J., vol. 16, no. 3, pp. 4629–4640, Sep. 2022, doi: 10.1109/JSYST.2021.3103909.
[20] P. Chaudhari and M. L. Das, ‘‘KeySea: Keyword-based search with receiver anonymity in attribute-based searchable encryption,’’ IEEE Trans. Services Comput., vol. 15, no. 2, pp. 1036–1044, Mar. 2022, doi: 10.1109/TSC.2020.2973570.M. Wang, Y. Guo, C. Zhang, C. Wang, H. Huang, and X. Jia, ‘‘MedShare: A privacy-preserving medical data sharing system by using blockchain,’’ IEEE Trans. Services Comput., vol. 16, no. 1, pp. 438–451, Jan. 2023, doi:
10.1109/TSC.2021.3114719.
[21] M. Ihtesham, S. Tahir, H. Tahir, A. Hasan, A. Sultan, S. Saeed, and O. Rana, ‘‘Privacy preserving and serverless homomorphic-based searchable encryption as a service (SEaaS),’’ IEEE Access, vol. 11, pp. 115204–115218, 2023, doi: 10.1109/access.2023.3324817.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 5, May 2025
Impact Factor: 7.67

[22] Y. Zhang, T. Zhu, R. Guo, S. Xu, H. Cui, and J. Cao, ‘‘Multi-keyword searchable and verifiable attribute-based encryption over cloud data,’’ IEEE Trans. Cloud Comput., vol. 11, no. 1, pp. 971–983, Jan. 2023, doi: 10.1109/TCC.2021.3119407.
[23] H. Li, Q. Huang, J. Huang, and W. Susilo, ‘‘Public-key authenticated encryption with keyword search supporting constant trapdoor generation and fast search,’’ IEEE Trans. Inf. Forensics Security, vol. 18, pp. 396–410, 2023, doi: 10.1109/TIFS.2022.3224308.
[24] J. Du, J. Zhou, Y. Lin, W. Zhang, and J. Wei, ‘‘Secure and verifiable keyword search in multiple clouds,’’ IEEE Syst. J., vol. 16, no. 2, pp. 2660–2671, Jun. 2022, doi: 10.1109/JSYST.2021.3069200.
[25] T. Liu, Y. Miao, K. R. Choo, H. Li, X. Liu, X. Meng, and R. H. Deng, ‘‘Time-controlled hierarchical multikeyword search over encrypted data in cloud-assisted IoT,’’ IEEE Internet Things J., vol. 9, no. 13, pp. 11017–11029, Jul. 2022, doi: 10.1109/JIOT.2021.3126468.
[26] F. Li, J. Ma, Y. Miao, Z. Liu, K. R. Choo, X. Liu, and R. H. Deng, Towards efficient verifiable Boolean search over encrypted cloud data,’’ IEEE Trans. Cloud Comput., vol. 11, no. 1, pp. 839–853, Jan. 2023, doi: 10.1109/TCC.2021.3118692.
[27] X. Li, Q. Tong, J. Zhao, Y. Miao, S. Ma, J. Weng, J. Ma, and K. R. Choo, data,’’ IEEE Trans. Services Comput., vol. 16, no. 1, pp. 698–710, Jan. 2023, doi: 10.1109/TSC.2021.3140092.
[28] X. Liu, X. Yang, Y. Luo, and Q. Zhang, ‘‘Verifiable multikeyword search encryption scheme with anonymous key generation for medical Internet of Things,’’ IEEE Internet Things J., vol. 9, no. 22, pp. 22315–22326, Nov. 2022, doi: 10.1109/JIOT.2021.3056116.
[29] H. Shen, L. Xue, H. Wang, L. Zhang, and J. Zhang, ‘‘B+-tree based multi-keyword ranked similarity search scheme over encrypted cloud data,’’ IEEE Access, vol. 9, pp. 150865–150877, 2021, doi: 10.1109/ACCESS.2021.3125729.
[30] Y. Zheng, R. Lu, Y. Guan, J. Shao, and H. Zhu, ‘‘Achieving efficient and privacy-preserving exact set similarity search over encrypted data,’’ IEEE Trans. Dependable Secure Comput., vol. 19, no. 2, pp. 1090–1103, Mar. 2022, doi: 10.1109/TDSC.2020.3004442.
[31] F. Li, J. Ma, Y. Miao, Q. Jiang, X. Liu, and K. R. Choo, ‘‘Verifiable and dynamic multi-keyword search over encrypted cloud data using bitmap,’’ IEEE Trans. Cloud Comput., vol. 11, no. 1, pp. 336–348, Jan. 2023, doi: 10.1109/TCC.2021.3093304.
[32] J. Shao, R. Lu, Y. Guan, and G. Wei, ‘‘Achieve efficient and verifiable conjunctive and fuzzy queries over encrypted data in cloud,’’ IEEE Trans. Services Comput., vol. 15, no. 1, pp. 124–137, Jan. 2022, doi: 10.1109/TSC.2019.2924372.
[33] X. Wang, J. Ma, X. Liu, Y. Miao, Y. Liu, and R. H. Deng, ‘‘Forward/backward and content private DSSE for spatial keyword queries,’’ IEEE Trans. Dependable Secure Comput., vol. 20, no. 4, pp. 3358–3370, Jul. 2023, doi: 10.1109/TDSC.2022.3205670.
[34] W. Rong-Bing, L. Ya-Nan, X. Hong-Yan, F. Yong, and Z. Yong-Gang, ‘‘Electronic scoring scheme based on real Paillier encryption algorithms,’’ IEEE Access, vol. 7, pp. 128043–128053, 2019, doi: 10.1109/ACCESS.2019.2939227.
[35] Y. Miao, R. H. Deng, K. R. Choo, X. Liu, and H. Li, ‘‘Threshold multi-keyword search for cloud-based group data sharing,’’ IEEE Trans. Cloud Comput., vol. 10, no. 3, pp. 2146–2162, Jul. 2022, doi: 10.1109/TCC.2020.2999775.
[36] Y. Lu and J. Li, ‘‘Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices,’’ IEEE Trans. Mobile Comput., vol. 21, no. 12, pp. 4397–4409, Dec. 2022, doi: 10.1109/TMC.2021.3077508.
[37] CSIRO’s Data61, GitHub Repository. (2013). Python Paillier Library. Accessed: Oct. 24, 2023. [Online]. Available: https://github.com/data61/python-paillier
[38] Ethereum, GitHub Repository. (2013). web3.py. Accessed: Oct. 24, 2023. [Online]. Available: https://github.com/ethereum/web3.py
[39] scikit-learn, Scikit-learn.org. (2019). Scikit-Learn: Machine Learning in Python. Accessed: Oct. 25, 2023. [Online]. Available: https://scikitlearn.org/stable/
[40] T. Zeutschler. (Jun. 30, 2023). TinyOlap. GitHub. Accessed: Oct. 7, 2023.