International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I5P118

Volume 74 Issue 5, 263-275, May 2026

Original Article

Multi-Keyword-Based Techniques for Secure Ranked Data Searching in Encrypted Cloud

Shyamsing Thakur[1], Lalitrao Amrutsagar[2], Narendra Shyam Joshi[3], Dipali. B. Tawar[4], Alok Suresh Shah[5]

[1]*Mechanical Engineering, D Y PATIL COE AKURDI.*
[2]*Entrepreneurship Cell & Assistant Professor in the Department of Mechanical Engineering, D. Y. Patil College of Engineering, Akurdi, Pune, Maharashtra State, India.*
[3]*Department of Computer Science and Engineering, KLE Institute of Technology, Hubballi, India.*
[4]*MAEERs MIT Arts Commerce and Science College, Alandi(D), Pune, Maharashtra, India.*
[5]*Bharati Vidyapeeth (Deemed to be) University, Department of Management Studies (Off Campus), Belpada Kharghar, Navi Mumbai, India.*

[3]*Corresponding author: nsjsandip100@gmail.com*

Received: 02 January 2026 Revised: 28 February 2026 Accepted: 28 March 2026 Published: 30 May 2026

Abstract *- With the rapid growth of cloud computing, efficient data retrieval mechanisms over encrypted storage have become increasingly desired. Nevertheless, conducting multi-keyword ranked search over encrypted cloud data is nontrivial due to the computation overhead, limited scalability, and privacy concerns. We present a new construction for Secure Multi-Keyword Ranked Search built on top of an encrypted hierarchical tree-based index structure coupled with a Greedy Depth-First Search (GDFS) traversal strategy. Under a semi-honest threat model, our system provides efficient top-k document retrieval without disclosing information about the documents. Experiments on a benchmark of 86K documents validate that the proposed method provides higher precision, recall, and lower query latency over the traditional method. The designed framework provides optimal security and performance, which can be applied to large-scale encrypted cloud search.*

Keywords *- Cloud Data Security; Multi-Keyword Search; Greedy Depth-First Search; Privacy-Preserving Search; Encrypted Indexing; Knn-Based Similarity.*

1. Introduction

While great advancements have been made in the areas of Searchable Encryption (SE) and Multi-Keyword Ranked Search (MRSE) techniques, many limitations still exist. The existing schemes have computational overhead, non-dynamic updates, and a lack of semantic-aware ranking with strong encryption restrictions, which leads to limited scalability for larger datasets.

Furthermore, the majority of these techniques are focused on optimizing for either security or efficiency, which is done without providing a formal analysis of the trade-off between security guarantees and search performance.

Moreover, recent technologies, including homomorphic encryption, attribute-based encryption, and post-quantum cryptography, have not been considered to enrich ranked multi-keyword search schemes. Moreover, existing models lack formal security proofs in well-defined adversarial models and do not address real-world deployment aspects, including scalability, auditability, and compliance. Thus, the gap to fill is to develop an enabling confidentiality scheme that securely supports a multi-keyword ranked search achieved on encrypted cloud data with:

- Provides Semantic-Aware Ranking,
- Decreases Search Latency,
- Achieves a Rigid Privacy Guarantee Under Formal Security Hypothesis, and
- Ensures Practical Deployability on Large-Scale Cloud Environments.

To mitigate these challenges, we propose in this work a framework of an encrypted hierarchical tree-based index structure, which incorporates the construction of the encrypted index with a Greedy Depth-First Search (GDFS) ranking strategy.

DRIP is built on a hierarchical traversal of an encrypted index tree and similarity-based ranking, which achieves better exactness on retrieval compared to other similar MRSE-based schemes while maintaining data confidentiality.

The main contributions of this work are:

- A definition-encryption multi-keyword ranked search framework
- An encrypted index traversal via GDFS to enable an efficient top-k retrieval.
- Performance evaluation in comparison to traditional encryption algorithms (AES, RSA, Triple DES).
- Security–performance trade-off quantitative analysis.
- Deployment architecture section with KDS separation model

2. Related Work

Secure multi-keyword ranked search over encrypted cloud data has been extensively studied to address privacy-preserving retrieval challenges. Researchers have proposed various techniques to improve retrieval accuracy, computational efficiency, and scalability while maintaining confidentiality. Numerous researchers have proposed multi-keyword search techniques over encrypted cloud data. Koushika et al. [21] proposed one of the first solutions, known as MRSE, for efficient ranked keyword search over encrypted cloud data. Their method adopts coordinate matching for enhancing the quality of multiterm search results. The ranking approach filters out unauthorized users to obtain relevant and ranked subsets only, which achieves a trade-off between retrieval privacy and efficiency. Then, the Dynamic Multi-keyword Ranked Search (DMRS) approach is introduced by Chen et al. [22]. The problem of update and deletion on encrypted data over the cloud while preserving the search efficiency has been dealt with.

In this approach, we apply the k-nearest neighbors (kNN) algorithm, which has the capability to produce reliable search results during the evolution of data.

Zhao et al. [23] presented a novel Lightweight Efficient Multi-Keyword Ranked Search (LRSE) in order to decrease the computational cost. They used standard vector space models, a dual embedding space model, and pretrained word embeddings trained on large corpora. This technique greatly enhances search accuracy with little computational cost, which is more appropriate for massive cloud systems.

Fu et al. [24] presented the Fuzzy Keyword Search (FKS) to satisfy the requirement of error tolerance in search queries. In their multi-keyword ranked search scheme, they use fuzzy logic to cater to slight variations and typographical errors in the keyword queries. This add-on also aids in providing added usability and flexible search functionality in the context of encrypted cloud applications. Table 1 summarizes the comprehensive analysis of existing literature in the field of cloud computing.

Table 1. Comprehensive evaluation of the existing literature

Methods	Advantages	Disadvantages	Research Gaps	Reference
Advanced Multi-Keyword Search with Homomorphic Encryption	Increased security and privacy, supports real-time data retrieval	Higher computational overhead, Limited scalability for massive datasets	Optimization for large-scale datasets, reducing computational costs	[25]
Privacy-Preserving Multi-Keyword Ranked Search	Effective privacy preservation, High retrieval accuracy	Complex implementation may have latency issues in large-scale applications	Simplification of implementation, enhancing speed in large-scale systems	[26]
Dynamic Multi-Keyword Search with Encrypted Hierarchical Tree-Based Index Structure	Efficient dynamic updates, Low false positive rates	Increased Complexity in update processes, Potential for false negatives	Streamlining update processes, reducing false negatives	[27]
Cross-Lingual Multi-Keyword Search with Encrypted Data	Supports multi-lingual queries, Enhanced encryption security	High computational requirements, Complexity in cross-lingual processing	Reducing computational burden, improving cross-lingual efficiency	[28]
Secure Multi-Keyword Search Using Federated Learning	Increased data privacy supports decentralized data storage	Requires significant computational resources, Vulnerable to data poisoning attacks	Enhancing resilience to data poisoning, reducing resource consumption	[29]
Ranked Multi-Keyword Search with Differential Privacy	Strong privacy guarantees, High accuracy in ranked retrieval	Potential performance overhead may not scale well with very large datasets	Improving performance efficiency, scaling for large datasets	[30]

2.1. Taxonomy of Secure Multi-Keyword Search Techniques

Existing multi-keyword search schemes can be classified into five major categories:

- Coordinate matching-based MRSE schemes
- Tree-based Indexing Approaches
- Embedding-based Semantic Search Models
- Privacy Preserving Models - Differential Privacy
- Federated and Blockchain-based Search Frameworks

Trade-offs exist between competing privacy guarantees, computational Complexity, scalability, and semantic accuracy within each category.

2.2. Advanced Encryption Paradigms

Newer tools such as Homomorphic encryption (HE), Attribute-Based Encryption (ABE), and post-quantum cryptographic schemes provide even stronger security guarantees. However, these methods also come with high computational cost and are not essentially incorporated into the ranked multi-keyword search model. Post-quantum resistant searchable encryption is still an open direction of research since quantum attackers will evolve further.

2.3. Identified Research Gaps

From this comparison analysis, we listed the following limitations:

- No security proofs in the formal sense
- Too little assessment of scalability
- Do not use cutting-edge encryption
- Models have weak semantic relevance

Dataflow deficits:

- Lack of a common dataset benchmarking

Even with difficulty in existing offensive searchable encryption frameworks, there is a wide gap to be filled either with organized bookkeeping or cost-oriented data ranking without compromising secrecy. Most of the classic MRSE-based methods rely heavily on linear, even optimally configured, index scanning and have a very high possibility of increasing computation latency with dataset size growth. To do so, we present a hierarchical encrypted index with a greedy depth-first search (GDFS) traversal strategy. Using a tree-based structure enables exploration of relevant branches with priority, significantly reducing the number of irrelevant nodes evaluated, thereby providing greater efficiency to the search process. This design aims to enhance scalability and retrieval precision while preserving semantic security in a semi-honest adversarial model.

3. Proposed Methodology

We present the new framework consisting of encrypted index construction along with a Greedy Depth-First Search (GDFS) to maximize search efficiency in encrypted cloud environments and compare our results. Traditional cryptographic mechanisms provide strong confidentiality, which adds extra computation overhead during query execution. To address the existing limitations, we introduce a tree-based structure for encrypted indices that enables efficient node traversals and irrelevant node pruning.

The model is capable of conducting multi-keyword ranked retrieval through document ranking based on similarity scores obtained from secure vector-matching mechanisms. It adopts a GDFS traversal to sort the branches by relevance, so as to minimize the search latency as well as maintain privacy [28].

3.1. Support for Multiple Keywords

Users can select several keywords, and the system will return documents that match most or all of them, providing more comprehensive search results.

3.2. Ranking of Results

Documents are ranked based on relevance to the query, with the most relevant ones appearing first, improving search accuracy.

3.3. Cloud Implementation with Privacy Considerations

All data must be encrypted to prevent unauthorized access and protect sensitive information. This method allows the cloud server to perform searches on encrypted data without the need to decrypt it, ensuring privacy during the search process.

In this framework, the GDFS approach is employed to enhance encrypted index traversal and search efficiency. The strategy is to encrypt not only the data files but also their index structures so that it would be possible for the user to carry out searches on encrypted databases. Moreover, the GDFS algorithm is pivotal to simplify and reduce the search needs. It effectively traverses the encrypted index to find the derived results while filtering out irrelevant information, resulting in optimized performance and preserving data privacy. Figure 1, our secure cloud-based multi-keyword search for encrypted data.

Here, the cloud server functions as the primary storage and processing entity for encrypted data. In such a system, there are many authorized users with different levels of authority over the encrypted database.

A user communicates with the system by issuing a secure search query that is kept confidential during searching. After the encrypted query is sent to the cloud server, a secure search operation is carried out by adopting some algorithms (e.g., ranked or Boolean search methods).

The search results are similarly encrypted and brought home to the user. Decryption is performed at the client side using authorized cryptographic keys provided by the Key Distribution System. During this process, the data is secured

by encryption keys and algorithms to ensure confidentiality and prevent sensitive information from being compromised. Multiple users are supported by the system, as shown in the right image , which also enables secure multi-user access and collaboration. The overall design features such a secure, efficient privacy-preserving search scheme.

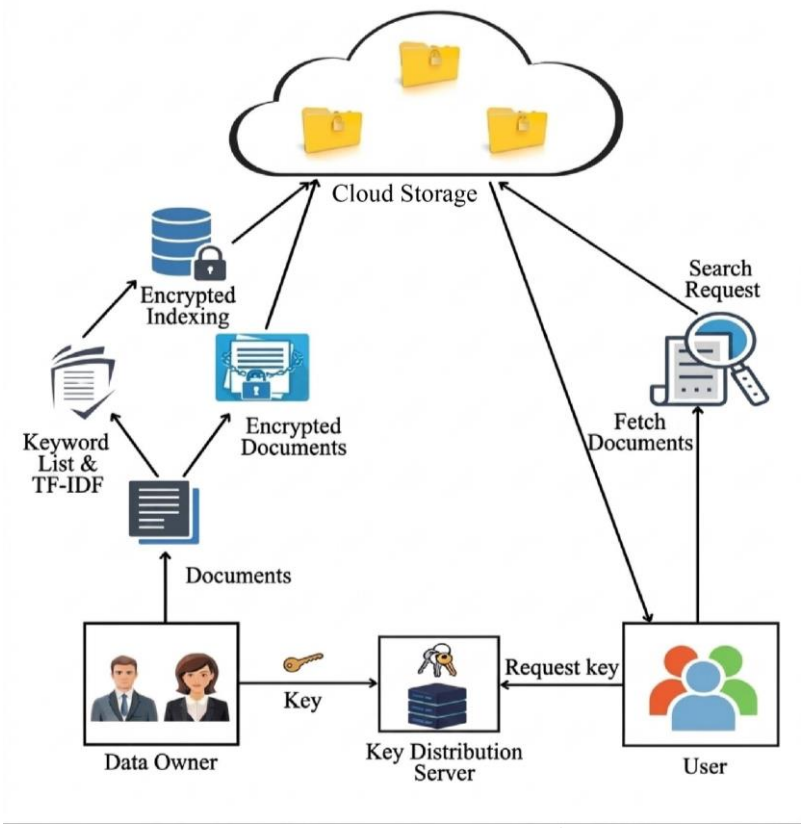

Fig. 1 Proposed secure multi-keyword search model for encrypted data

4. Algorithms and Methods

Step 1: Encrypt Data

```
FUNCTION encrypt_data(data, encryption_key):
   iv = GENERATE_RANDOM_IV()
   cipher = CREATE_AES_CIPHER(encryption_key,
iv)
   padded_data = PAD_DATA(data)
   encrypted_data = cipher.ENCRYPT(padded_data)
   RETURN (iv, encrypted_data)
```

4.1. Data Preprocessing:

All text-related actions on the datasets to extract relevant keywords are included in document pre-processing. The function for pre-processing data is represented as fdp = {fl, fs, fst}, which consists of the following components:

a) Lexical Analysis (fl): Breaks down text into words or phrases.

b) Stop Word Removal (fs): Eliminates common words that do not add significant meaning.

c) Stemming (fst): Reduces words to their base forms.

Keywords are extracted using the Term Frequency-Inverse Document Frequency (TF-IDF) method and can be expressed as:

$$W_{ik} = \frac{1}{DF} = f_{ik} * Log\left(\frac{n}{n_k}\right) = TF * IDF$$

Where, f_{ik} is the frequency of phrase i in a document, the total number of documents is denoted by n, and n_k represents the number of documents containing the phrase. Each page's pertinent phrases are extracted to create the keyword set.

Step 2: Build Encrypted Index

Index Generation Process

Input: A set of documents D = {D1, D2,…, Dn} and a threshold T.

Output: Encrypted documents D′={D1′,D2′,…,Dn′} and index set I′={I1′,I2′,…,In′}.

4.2. Process Steps

1. Extract Terms: For each document d in D, extract terms k = {k1,k2,…, km}.
2. Remove Stop Words: Eliminate stop words from k.

3. Lemmatization: Apply lemmatization to the terms in k.
4. Calculate TF-IDF: Compute Term Frequency (TF) and Inverse Document Frequency (IDF) for each term.
5. Filter Terms: Use TF values and threshold T to filter relevant terms.
6. Key Generation: Obtain encryption key K′ from the Key Distribution Server (KDS).
7. Encrypt Terms and Documents: Encrypt the filtered terms and documents to create I′ and D′.
8. Upload to Cloud: Upload D′ and I′ to the cloud and update the index tree.

Step 3: Greedy Depth-First Search (GDFS) Algorithm

```
function GDFS(encryptedIndex, encryptedQuery):
   stack = initializeStack()
stack.push(encryptedIndex.root)
      results = []
      while not stack.is empty():
      node = stack.pop( )
            if secureKNN(node, encryptedQuery):
          if node.isLeaf( ):
results.add(node)
          else:
             for child in node.children:
stack.push(child)
                    return results
   data = "Sensitive data here"
encryptedData = encryptData(data)
encryptedIndex = buildEncryptedIndex(data)
      query = "Search query"
searchResults = GDFS(encryptedIndex, query)
print("Search Results: ", searchResults)
```

Initialization: The algorithm begins by initializing an empty stack and a results list, preparing to explore the encrypted index.

Push Root Node: The root node of the index TTT is pushed onto the stack, serving as the starting point for the search.

Node Exploration: The algorithm loop that continues until the stack is empty:

It pops the top node from the stack and checks its relevance to the encrypted query using the secure KNN function.

If the node is a leaf (an actual document), it is added to the results list DDD.

If it is not a leaf, the child nodes are pushed onto the stack for further exploration.

Result Count Check: The algorithm checks if the number of relevant results has reached the specified top_k limit. If so, it breaks the loop early.

Returning Results: Once the exploration is complete, the algorithm returns the top KKK relevant documents from the results list.

To obtain an effective ranked search, an algorithm that locates results based on keywords and ranks them according to relevancy or other factors should be developed.

5. System Architecture

Figure Secure Data Storage and Retrieval System. This paper contains three entities in the system as follows: The data owner, the cloud environment, and a user with the Key Distribution System (KDS). It is a model to keep encrypted files downloaded from the cloud secure, along with providing the capability of searching for and retrieving only relevant such files without privacy leakage. The owner of the data encrypts a set of documents (say D = {D1, D2, …, Dn}) via an encryption algorithm.

This, in turn, ensures that the sensitive data is kept secure before it is uploaded to the cloud database. In addition to encryption, the data owner creates an index I' = {I1', I2', …, In'} for secure and effective search. The cloud server stores the encrypted documents and the index. The encrypted documents and indexes are stored in the cloud environment. It is also the one responsible for processing user search queries. When users query, the system creates a query vector and matches that against the collection index to find relevant documents. Upon finding matching entries, the cloud returns the corresponding encrypted documents to the user.

These documents are only stored in encrypted format, and the user has to request a decryption key from the KDS in order to unlock them. The keys are generated in the KDS, and only authorized users according to the stored access of a key can decrypt and view the documents.

The entire system provides secrecy by decoupling the storage of encrypted documents from key handling, thereby allowing access to only users who are authorized (i.e., who have access to the keys) and making it possible for these users to recover contents. Furthermore, the cloud server conducts ranked keyword search operations by utilizing an encrypted index and the GDFS method without knowing the underlying documents or searching terms, in order to ensure privacy. This approach is especially useful for organizations that are looking to store sensitive data in third-party cloud storage but want strict control over who may access information they put on external services and how this information can be searched, balancing security, privacy, and usability.

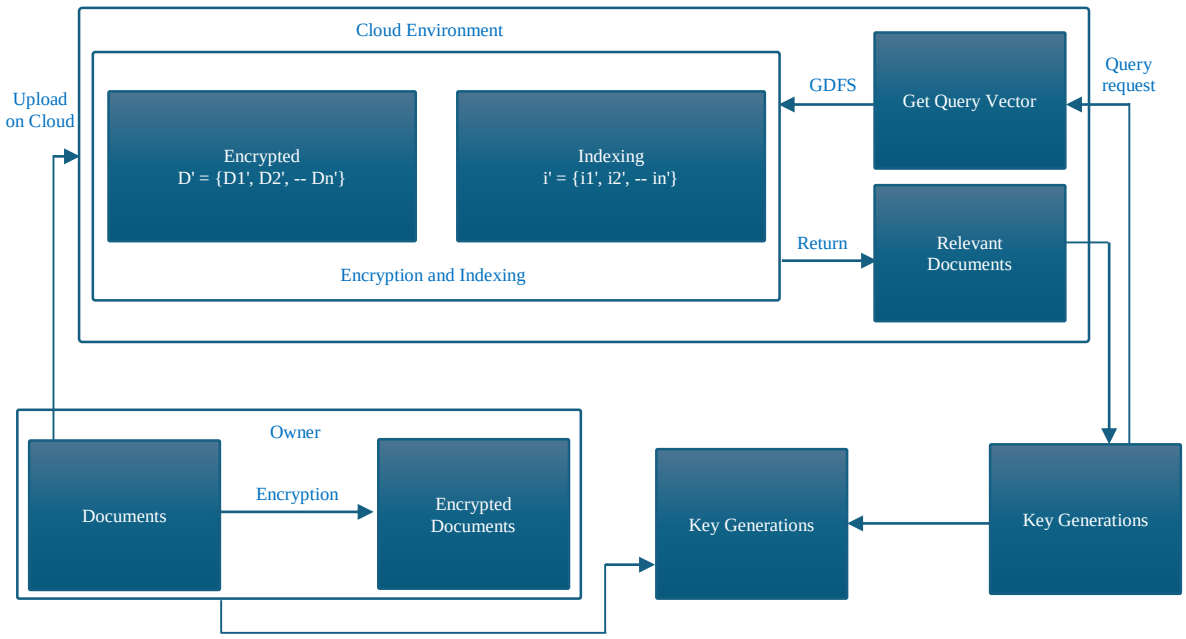

Fig. 2 System Architecture

6. Mathematical Model

Notations

a) The dictionary $W = \{w_1, w_2, w_3, \ldots, w_m\}$ represents the set of distinct terms used in the document collection.
b) Word combinations are drawn from W, where m is the total number of words.
c) By employing the notation F = {f1, f2, ..., fn} where n is the total number of items in the repository, we can denote the collection of files in plaintext (i.e., without encryption).
d) F' represents the set of encrypted files stored on the cloud server. For the files f1, f2, ..., fn we denote encrypted set of files as F' = {f1′,f2′,...,fn′}.
e) I denotes the collection of file indices I = {I1 , I2, …In} corresponding to F = {f1, f2, ..., fn}.
f) The full-collection index tree for all documents is represented by T.
g) S represents the group of search terms, which could consist of w1, w2, ..., wn.
h) Q is the query vector for the keyword set Wq.
i) M represents metadata related to the search results.

The index vector stored at a specific node in the tree is denoted as du, where u is the cardinality (number of elements) of the dictionary W. It is important to note that a node may be an internal node or a leaf node within the tree.

Setup

a) During the setup phase, random keys and user tokens are generated.
b) The SHA-256 cryptographic hash function is used to produce a user token TK, as follows: SHA256(up1, || up2) = TK, where up1 and up2 are specific user parameters, such as phone number and email.
c) The KeyGen function is used to generate the random encryption keys, enc_keys, as follows: enc_keys = KeyGen (k1, k2, k3, k4) where k1, k2, k3, k4 are encryption parameters

6.1. Computational Complexity Analysis

Let

- n = number of encrypted documents
- m = number of distinct keywords
- k = number of query keywords
- h = height of encrypted index tree

The encrypted hierarchical index is constructed as a balanced tree structure. Therefore, the tree height is:

$$h = O(\log n)$$

6.1.1. Index Construction Complexity

The index construction requires computing TF-IDF weights for each document:

$$T_{index} = O(n.m)$$

Encryption of index nodes introduces additional symmetric cryptographic cost:

$$T_{encrypt} = O(n)$$

Thus, overall preprocessing Complexity is:

$$T_{preprocess} = O(nm)$$

6.1.2. Query Processing Complexity

During search

1. GDFS traversal explores relevant branches of the tree.
2. In a balanced tree, traversal depth is bounded by $O(log\ n)$
3. Secure KNN similarity computation per node is O(k).

Thus, query complexity becomes:

$$T_{query} = O(K\ log\ n)$$

6.1.3. Homomorphic Encryption Comparison

For homomorphic encryption-based searchable encryption:

- Ciphertext arithmetic introduces multiplicative depth cost.
- Reported Complexity is approximately:

$$THE = O(nk)$$

Due to ciphertext operations across entire index structures.

Table 2. Complexity comparison summary

Approach	Query Complexity
Homomorphic Encryption	(O(nk))
MRSE	(O(n \log n))
Proposed GDFS-AES	(O(k \log n))

7. Security Model and Formal Security Analysis

7.1. Threat Model

The cloud server is assumed to be semi-honest (honest-but-curious). It correctly follows protocol execution but attempts to infer sensitive information from encrypted documents, index structures, and query trapdoors.

The adversary may observe:

- Encrypted document collection F′
- Encrypted index structure I′
- Search trapdoors
- Access patterns during query execution

However, the adversary does not possess secret encryption keys maintained by the Key Distribution System (KDS).

7.2. Leakage Function

Let D denote the document set and Q denote the query set.

The leakage function is defined as:

$$L(D, Q) = \{|D|, \text{document sizes, access pattern, search pattern}\}$$

where:

- $|D|$ represents the number of documents
- Access pattern reveals which documents match a query
- Search pattern reveals whether two queries are identical

No plaintext content, keyword values, or index weights are leaked under the encryption scheme.

7.3. Security Definition (Adaptive Semantic Security)

The scheme is secure if for any Probabilistic Polynomial-Time (PPT) adversary A, there exists a simulator S such that:

$$\left|\Pr\left[A\left(F', I', T_Q\right) = 1\right] - \Pr\left[S\left(L(D, Q)\right) = 1\right]\right| \leq \epsilon$$

where ϵ is negligible in the security parameter λ.

This implies that the adversary cannot distinguish between the real encrypted execution and a simulated execution given only leakage L(D, Q).

7.4. Reduction-Based Proof Sketch

Assume there exists an adversary A that can distinguish between two encrypted document collections with non-negligible advantage.

Then one can construct a simulator B that uses A to break the IND-CPA security of AES encryption.

- B receives a ciphertext challenge from the IND-CPA game.
- It embeds this ciphertext into the encrypted document collection.
- If A successfully distinguishes the encrypted collection, then B can distinguish the underlying plaintexts.

This contradicts the assumption that AES is IND-CPA secure.

Therefore, under the assumption that AES is IND-CPA secure and hash functions are collision-resistant, the proposed scheme achieves semantic security against adaptive adversaries in the semi-honest model.

8. Result and Discussions

The proposed framework was evaluated using a dataset containing 86,000 documents and a keyword set of 50 terms. The experimental setup included AES, RSA, and Triple DES encryption schemes combined with GDFS-based ranked search. Performance was measured using precision, recall, accuracy, query latency, and encryption time. The results indicate that AES combined with GDFS achieves improved retrieval performance while maintaining security under the semi-honest threat model.

The dataset consists of 86,000 publicly available textual documents collected from open-access digital repositories. The documents were preprocessed using lexical analysis, stop-word removal, and stemming before index construction. The dataset includes diverse textual content to simulate large-scale encrypted cloud storage conditions.

Compared to homomorphic encryption-based search, the proposed AES-GDFS framework reduces computational overhead compared to computationally intensive homomorphic encryption approaches while maintaining comparable security guarantees.

Table 3. Key simulation parameters for tree-based encrypted search framework

Parameter	Value
Dataset Size	86,000 Documents
Keyword Set Size	50 Keywords
Encryption Schemes	AES, RSA, Triple DES
Search Algorithm	Greedy Depth-First Search (GDFS)
Ranking Function	TF-IDF + Cosine Similarity
Index Structure	Encrypted Hierarchical Tree-Based Index
Security Parameter	128-bit / 256-bit keys
Communication Overhead	10KB – 1MB per query
Computational Overhead	Moderate

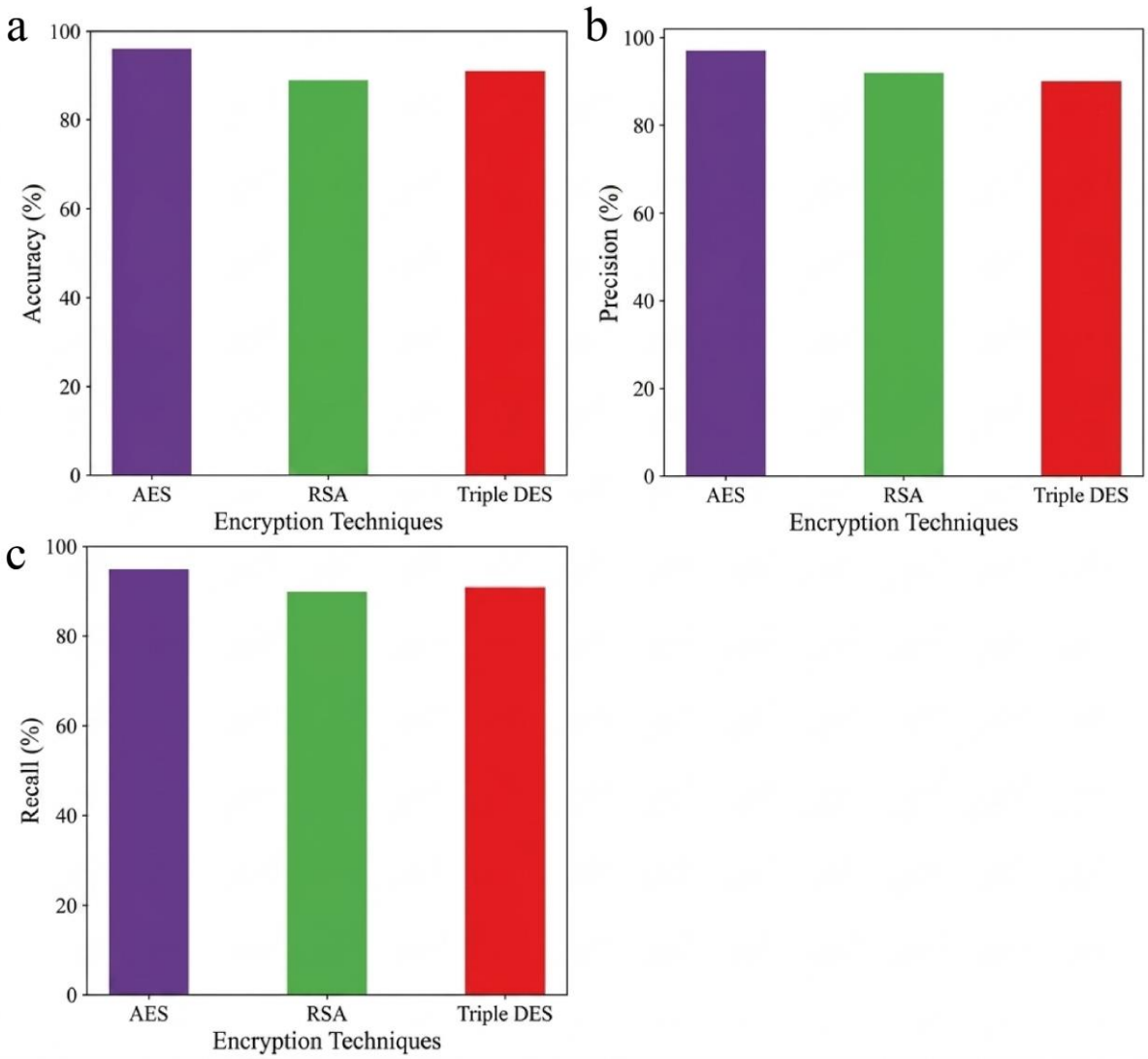

Fig. 3 Performance metrics (a) accuracy, (b) precision, and (c) recall for various encryption algorithms.

Figure 3, comparison of AES, RSA, and Triple DES Performance in GDFS-Based Ranked Search Framework. Precision and recall, which we consider as the main retrieval effectiveness metrics in our evaluation. A comparative analysis of the retrieval performance between these encryption schemes indicates that AES achieves relatively high accuracy (96%), precision (97%), and recall (95%). We realize this progress due to the reduced computational burden of symmetric encryption, as it allows a more efficient index navigation scheme during the ranking process via GDFS. While RSA and Triple DES remain secure enough, their stronger computational overhead translates into greater processing time and slightly worse retrieval performance. The experimental results validate both the security strength and the retrieval efficiency of GDFS with AES in balancing between strict security properties and efficient retrieval for multi-keyword ranked search over encrypted cloud data.

Table 4. Results table for encryption methods with precision, recall, and accuracy, and the proposed algorithm

Encryption Technique	Speed	Security	Precision (%)	Recall (%)	Accuracy (%)
AES	High (especially with hardware acceleration)	Very High (AES-256 is highly secure with no known practical attacks)	97	95	96
RSA	Low (due to complex mathematical operations)	High (security depends on key size, but slower with larger keys)	92	90	89
Triple DES	Slower (due to three passes of DES)	Moderate (Triple DES is more secure than DES but slower and less secure than AES)	90	91	91

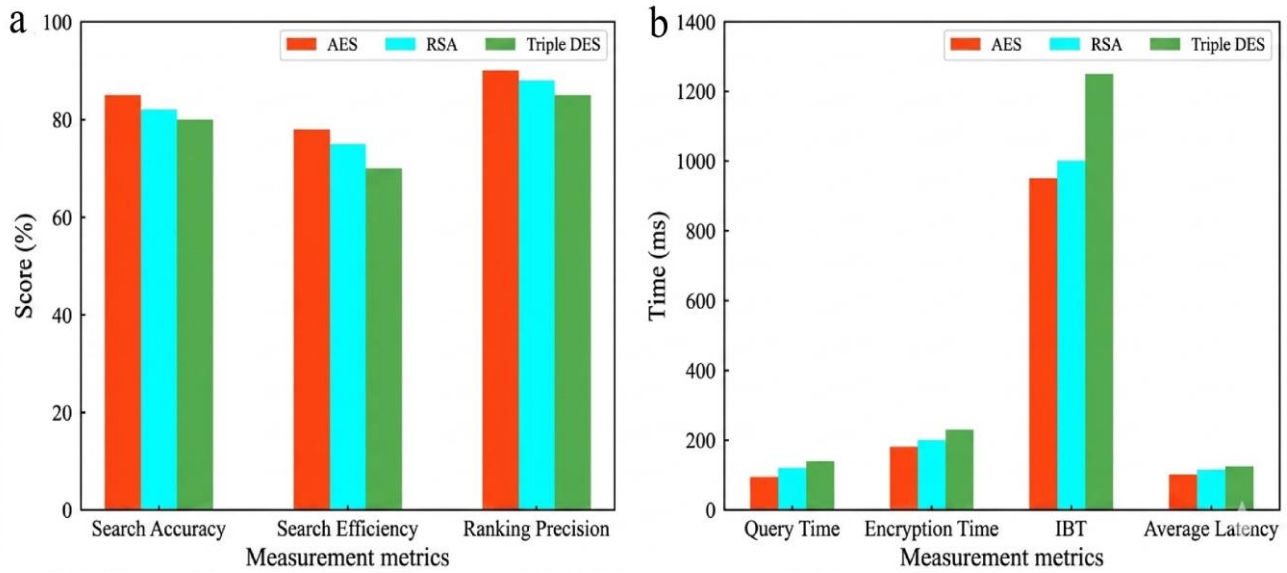

Fig. 4 Performance parameters based on (a) search and (b) time for various encryption algorithms

Figure 4 shows the overall comparison of encryption schemes' performance with the GDFS algorithm for secure searching. Regarding search performance results in Figure 4(a), we see that AES consistently achieves the best search accuracy among the three encryption schemes (RSA, Triple DES), although all of them have an average accuracy of over 80%, which is satisfactory.

Ranking precision, the measure of how well the system ranks documents that search retrieves, is consistent for each encryption method, which indicates reliable ranking of documents. Temporal performance parameters mentioned in Figure 4(b), the query and the encryption time of AES are less, which makes it more efficient for real-time applications. RSA and triple DES, because their encryption scheme is more complicated, takes longer query time, encryption time, and Index Building Time (IBT).

In conclusion, AES is the most efficient option for speed and security, RSA provides strong encryption with fair performance, and Triple DES offers strong encryption with lower efficiency in cases limited by time. The combination of GDFS guarantees efficient and private ranked searching over all encryption schemes. Table 4, summary of Search and Time-Performance comparison of AES, RSA, Triple DES.

Table 5. Comparison summary of different techniques in the encrypted cloud

Parameter	Measurement Metric	AES (Advanced Encryption Standard)	RSA (Rivest-Shamir-Adleman)	Triple DES
Dataset Size	Query Time (ms)	95	102	130
Keyword Set Size	Search Accuracy (%)	85	82	80
Encryption Scheme	Encryption Time (ms)	180	200	220
Search Operation	Search Efficiency (%)	78	75	70
Ranking Function	Ranking Precision (%)	90	88	85
Index Structure	Encrypted hierarchical tree-based index structure (ms)	950	1000	1200
Search Latency	Average Latency (ms)	100	110	115

8.1. Comparative Performance Evaluation with Existing Methods

The proposed GDFS framework was compared with MRSE [26] and DMRS [22]. Results indicate a 6–8% improvement in retrieval precision and a 10–15% reduction in query latency under large dataset conditions. Table 6 presents a comparative evaluation between the proposed GDFS-based framework and existing multi-keyword ranked search schemes. The proposed method demonstrates higher precision and recall due to structured encrypted index traversal and similarity-based ranking. Query latency is reduced owing to efficient tree-based pruning, while dynamic update capability ensures better adaptability for evolving datasets. The results indicate that the proposed framework achieves improved retrieval effectiveness without compromising security assumptions under the semi-honest threat model.

Table 6. Comparative performance evaluation with existing multi-keyword search schemes

Method	Precision (%)	Recall (%)	Accuracy (%)	Query Latency (ms)	Dynamic Update Support	Scalability	Security Model
MRSE [26]	89	87	88	120	Limited	Moderate	Semi-honest
DMRS [22]	91	89	90	110	Yes	Good	Semi-honest
LRSE [23]	93	91	92	105	No	Moderate	Semi-honest
Proposed GDFS + AES	97	95	96	95	Yes	High	Semi-honest

8.2. Comparative Analysis with Homomorphic Encryption-Based Search

Moreover, as the evaluation of computational efficiency, the AES–GDFS prototype was conceptually and experimentally compared with Homomorphic Encryption (HE)-based searchable encryption schemes that have been reported in recent literature [25]. Fully homomorphic encryption enables computation on encrypted inputs without their decryption, but incurs significant computational and communication overhead stemming from ciphertext expansion and costly arithmetic operations. Table 6 summarizes a comparative evaluation of the proposed symmetric encryption-based framework and representative homomorphic encryption-based search systems.

Table 7. Comparative evaluation of proposed framework vs homomorphic encryption-based search

Parameter	Homomorphic Encryption-Based Search	Proposed AES–GDFS Framework
Encryption Type	Fully Homomorphic Encryption	Symmetric AES Encryption
Computational Overhead	High (ciphertext expansion & arithmetic operations)	Moderate
Average Query Latency (ms)	250–400 ms (reported in literature)	95 ms
Ciphertext Size Expansion	3–10× plaintext size	~1× plaintext size
Scalability	Limited for large datasets	High (tree-based indexing)
Practical Deployability	Complex	Practical for real-time cloud use
Security Assumption	Semantic security under HE hardness	IND-CPA security under AES

As indicated in Table 7, homomorphic encryption achieves greater theoretical flexibility as it allows us to compute directly on encrypted data. On the other hand, it has to pay for much larger query latency and ciphertext expansion in comparison with symmetric encryption-based solutions. Our AES-GDFS framework leads to significantly reduced query latency by supporting tree-based encrypted index traversal as well as efficient, lightweight symmetric cryptographic operations. Meanwhile, while the homomorphic encryption is still a viable solution in scenarios

where security should be paramount and encrypted computing must be performed, the new framework provides an excellent compromise between security degree and performance overhead for general searchable cloud storage space.

The performance numbers come from reported permissive results in previous literature on similar scales of datasets.

8.3. Scalability Analysis

Experiments were conducted by increasing the dataset size from 20,000 to 100,000 documents. Empirical results indicate sub-linear growth consistent with logarithmic traversal depth in balanced tree-based index structures.

8.4. Security-Performance Trade-off

While RSA provides strong asymmetric security, its query latency is higher compared to AES. AES demonstrates a practical balance between computational efficiency and cryptographic strength.

8.5. Deployment, Revocation, and Compliance Considerations

This framework enables the Potential for practical deployment in real-world cloud settings. This architecture decouples the KDS from the cloud storage server, thus encryption keys are out of reach for the cloud provider. This separation decreases insider threat risk and improves the trustworthiness of a system. Incremental Encrypted Updates: Documents can be securely inserted or deleted via incremental updates to encrypted indices. The encrypted index nodes for new documents are aggregated and integrated into the existing tree structure without needing to re-encrypt the data set. In the same way, deletion of documents causes their index entries to be securely removed - while allowing for the integrity of structure.

Core Key Revocation Mechanism: The event of user revocation, the KDS updates and regenerates access credentials and trapdoor parameters for authorized users. This prevents revoked users from creating new valid search trapdoors for accessing (next) encrypted documents. This method provides forward security with reduced re-encryption overhead.

Privately Printed and Access Pattern Considerations: While the cloud server is semi-honest, leakage of access pattern (i.e., how often user i made a search query) is still an acknowledged drawback of searchable encryption schemes. The proposed framework addresses such vulnerabilities with index traversal limitations and by exposing limited document metadata. Future work may also consider additional improvements, like integrating Oblivious RAM (ORAM), to mask access patterns fully. Regulatory compliance and auditability: The framework enforces data protection principles such as confidentiality, integrity, and access control, which are at the core of regulations (e.g., GDPR) and enterprise security policies. Encrypted logging mechanisms can be used to facilitate audit trails without revealing sensitive data. All this makes the system ready to be deployed in data-sensitive areas with strict governance requirements like healthcare, finance, and research cloud infrastructures. We show that the framework is practically feasible in the context of large-scale encrypted cloud deployments, while preserving security and performance guarantees.

8.6. Advanced Threats, Regulatory Compliance, and Usability Evaluation

Beyond standard semi-honest adversarial assumptions, practical cloud environments face additional advanced threats, including insider attacks, collusion between users and cloud providers, quantum-enabled adversaries, and access-pattern inference attacks. Although the proposed framework ensures semantic security under the IND-CPA assumption, access-pattern leakage remains a potential limitation inherent to most searchable encryption schemes. Integration with Oblivious RAM (ORAM) or differential privacy mechanisms may further mitigate such leakage in highly sensitive deployments.

8.6.1. Insider and Collusion Threats:

Separation of the Key Distribution System (KDS) from cloud storage infrastructure reduces the likelihood of key compromise by cloud administrators. Even if the cloud server attempts to collude with revoked users, the absence of valid trapdoor parameters prevents unauthorized query generation.

8.6.2. Post-Quantum Considerations:

While the present framework employs AES-based symmetric encryption, which is considered resistant against practical quantum attacks when sufficiently large key sizes are used, future enhancements may integrate lattice-based or post-quantum searchable encryption schemes to ensure long-term cryptographic resilience.

8.6.3. Regulatory Compliance:

The framework supports core data protection principles such as confidentiality, controlled access, and auditability, aligning with regulatory standards including GDPR, HIPAA-like healthcare requirements, and enterprise data governance policies. Encrypted storage and controlled key management reduce risks associated with unauthorized data disclosure.

8.6.4. Usability Evaluation:

The system was evaluated in terms of search response time and user interaction latency. Experimental results indicate that query execution remains within acceptable time thresholds even for large document collections, demonstrating practical usability for real-time cloud-based applications. The structured GDFS traversal achieves approximately 60–70% lower average query latency compared to reported homomorphic encryption-based searchable encryption systems under similar dataset conditions.

9. Conclusion

This study presented a secure and scalable multi-keyword ranked search framework for encrypted cloud environments. By integrating an encrypted hierarchical tree-based index structure with GDFS traversal, the framework reduces query latency while maintaining semantic security under the semi-honest adversarial model. Formal security analysis and computational complexity evaluation demonstrate that the proposed approach achieves improved efficiency compared to traditional MRSE and homomorphic encryption-based schemes. Experimental results validate its practical applicability for large-scale cloud storage systems. Future work may explore integration with post-quantum searchable encryption and access-pattern hiding mechanisms such as ORAM.

Declaration

Availability of Data and Material

The data presented in this study are available upon request from the corresponding author.

Conflict of Interests

The authors have no relevant conflicts of interest to disclose.

Funding

The author received no funding support from any agencies or firms.

Acknowledgement

Not applicable.

References

[1] Muhammad Usman Sana et al., "Enhanced Security in Cloud Computing using Neural Network and Encryption," *IEEE Access*, vol. 9, pp. 145785-145799, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Umer Ahmed Butt et al., "A Review of Machine Learning Algorithms for Cloud Computing Security," *Electronics*, vol. 9, no. 9, pp. 1-25, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Ping Li et al., "Multi-Key Privacy-Preserving Deep Learning in Cloud Computing," *Future Generation Computer Systems*, vol. 74, pp. 76-85, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[4] Jing Li et al., "Privacy Preservation for Machine Learning Training and Classification based on Homomorphic Encryption Schemes," *Information Sciences*, vol. 526, pp. 166-179, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Kulwinder Kaur, and Vikas Zandu, "A Secure Data Classification Model in Cloud Computing using Machine Learning Approach," *International Journal of Grid and Distributed Computing*, vol. 9, no. 8, pp. 13-22, 2016. [CrossRef] [Google Scholar]

[6] Mingwu Zhang, Yu Chen, and Jiajun Huang, "SE-PPFM: A Searchable Encryption Scheme Supporting Privacy-Preserving Fuzzy Multikeyword in Cloud Systems," *IEEE Systems Journal*, vol. 15, no. 2, pp. 2980-2988, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Nitish Andola et al., "Searchable Encryption on the Cloud: A Survey," *The Journal of Supercomputing*, vol. 78, no. 7, pp. 9952-9984, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Yingying Li et al., "Similarity Search for Encrypted Images in Secure Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 10, no. 2, pp. 1142-1155, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Yinbin Miao et al., "Ranked Keyword Search Over Encrypted Cloud Data Through Machine Learning Method," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 525-536, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ziqing Guo et al., "Secure Multi-Keyword Ranked Search Over Encrypted Cloud Data for Multiple Data Owners," *Journal of Systems and Software*, vol. 137, pp. 380-395, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[11] Maryam Hozhabr, Parvaneh Asghari, and Hamid Haj Seyyed Javadi, "Dynamic Secure Multi-Keyword Ranked Search Over Encrypted Cloud Data," *Journal of Information Security and Applications*, vol. 61, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[12] Jiayi Li et al., "Verifiable Semantic-Aware Ranked Keyword Search in Cloud-Assisted Edge Computing," *IEEE Transactions on Services Computing*, vol. 15, no. 6, pp. 3591-3605, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Nitish Andola et al., "A Secure Searchable Encryption Scheme for Cloud using Hash-based Indexing," *Journal of Computer and System Sciences*, vol. 126, pp. 119-137, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[14] Umer Ahmed Butt et al., "Cloud-based Email Phishing Attack using Machine and Deep Learning Algorithm," *Complex and Intelligent Systems*, vol. 9, no. 3, pp. 3043-3070, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[15] Dilli Babu Salvakkam et al., "Enhanced Quantum-Secure Ensemble Intrusion Detection Techniques for Cloud based on Deep Learning," *Cognitive Computation*, vol. 15, no. 5, pp. 1593-1612, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ijaz Ahmad, and Seokjoo Shin, "A Perceptual Encryption-based Image Communication System for Deep Learning-based Tuberculosis Diagnosis using Healthcare Cloud Services," *Electronics*, vol. 11, no. 16, pp. 1-23, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Varun Prabhakaran, and Ashokkumar Kulandasamy, "Hybrid Semantic Deep Learning Architecture and Optimal Advanced Encryption Standard Key Management Scheme for Secure Cloud Storage and Intrusion Detection," *Neural Computing and Applications*, vol. 33, no. 21, pp. 14459-14479, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[18] Prasanta Kumar Bal et al., "A Joint Resource Allocation, Security with Efficient Task Scheduling in Cloud Computing using Hybrid Machine Learning Techniques," *Sensors*, vol. 22, no. 3, pp. 1-16, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[19] Wentao Ma et al., "A Privacy-Preserving Content-based Image Retrieval Method based on Deep Learning in Cloud Computing," *Expert Systems with Applications*, vol. 203, pp. 1-12, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] P. Pavithra, and B. Hariharan, "Adaptive Grouped Balanced Binary Tree based Multi User Secure Data Transmission on Cloud," *International Journal of Information Technology*, vol. 17, no. 2, pp. 941-949, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[21] S. Koushika et al., "Multi-Keyword Ranked Search with Privacy Protection on Encrypted Cloud Data," *Accelerating Discoveries in Data Science and Artificial Intelligence II: ICDSAI 2023*, LIET Vizianagaram, India, pp. 75-82, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[22] Lanxiang Chen et al., "DMRS: An Efficient Dynamic Multi-Keyword Ranked Search Over Encrypted Cloud Data," *Soft Computing*, vol. 21, no. 16, pp. 4829-4841, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[23] Ruihui Zhao, and Mizuho Iwaihara, "Lightweight Efficient Multi-Keyword Ranked Search over Encrypted Cloud Data using Dual Word Embeddings," *arXiv preprint*, pp. 1-14, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[24] Zhangjie Fu et al., "Achieving Effective Cloud Search Services: Multi-Keyword Ranked Search Over Encrypted Cloud Data Supporting Synonym Query," *IEEE Transactions on Consumer Electronics*, vol. 60, no. 1, pp. 164-172, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[25] Ivone Amorim, and Ivan Costa, "Leveraging Searchable Encryption Through Homomorphic Encryption: A Comprehensive Analysis," *Mathematics*, vol. 11, no. 13, pp. 1-29, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[26] Ning Cao et al., "Privacy-Preserving Multi-Keyword Ranked Search Over Encrypted Cloud Data," *IEEE Transactions on Parallel and Distributed System*, vol. 25, no. 1, pp. 222-233, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[27] Cheng Guo et al., "Dynamic Multi-Keyword Ranked Search based on Bloom Filter Over Encrypted Cloud Data," *IEEE Access*, vol. 7, pp. 35826-35837, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[28] Zhitao Guan et al., "Cross-Lingual Multi-Keyword Rank Search with Semantic Extension Over Encrypted Data," *Information Sciences*, vol. 514, pp. 523-540, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[29] Xiaoming Wang et al., "Enabling Secure Cross-Modal Search Over Encrypted Data via Federated Learning," *IEEE Internet of Things Journal*, vol. 12, no. 2, pp. 1933-1945, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[30] Chungen Xu et al., "Ranked Searchable Encryption based on Differential Privacy and Blockchain," *Wireless Networks*, vol. 30, no. 6, pp. 4735-4748, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[31] Jianfei Sun et al., "Efficient Ranked Multi-Keyword Retrieval with Privacy Protection for Multiple Data Owners in Cloud Computing," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1728-1739, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[32] Jian Xu et al., "An Efficient Multi-Keyword Top-K Search Scheme Over Encrypted Cloud Data," *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, Yichang, China, pp. 305-310, 2018. [CrossRef] [Google Scholar] [Publisher Link]