Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

CryptoCloud: A Usable Framework for Storing, Searching and Sharing Encrypted Data on Cloud Storage

Md. Mazharul Islam[1] | **Md. Shahadul Alam Patwary**[1] | **Salekul Islam**[1] | **Rajesh Palit**[1]

[1] Department of Electrical and Computer Engineering, North South University, Dhaka, Bangladesh

Correspondence

Corresponding author: Salekul Islam.
Email: salekul.islam@northsouth.edu

Present address

Dhaka, Bangladesh.

Abstract

Security has become a significant concern with the increased popularity of cloud storage services. It comes with the vulnerability of being accessed by third parties. It has given rise to security concerns in data confidentiality and unauthorized access, even after the deletion of data from cloud storage. This becomes a severe issue when the replicated copies remain hidden in the server even after the owner has deleted the original file. Thus, data needs to be encrypted before uploading to the public cloud. However, it creates a hindrance when the encrypted data needs to be shared with third parties without any already established trusted relationship. Besides, searching, a crucial functionality in cloud storage, does not work on any encrypted data. Searchable encryption (SE) allows a cloud server to conduct a search over encrypted data on behalf of the data users without learning the stored data. While many SE schemes show provable security, they expose query information and do not provide effective data utilization. Also, sharing encrypted data with other authorized users must ensure providing a document-specific secret key, creating massive key management overhead. To address the above issues, we design CryptoCloud, a secure and searchable encryption framework with an inverted index using symmetric cryptography with Identity-Based Encryption (IBE). We implement the system using widely used cryptographic techniques, ensuring search on encrypted data. The primary focus is to ensure user privacy and security through a less computationally intensive, user-friendly system with a trusted third-party entity. The system authenticates users through this entity and issues encrypted search keywords, ensuring identical queries remain unlinkable to the cloud. Its encrypted inverted index supports efficient single and multi-keyword search, while avoiding per-document key distribution to minimize the key-management overhead. All the cryptographic operations are performed at the trusted third-party and client sides, so plaintext and keys are never exposed to the storage provider. We also develop a web application as an overlay system of CryptoCloud on a cloud storage domain. Our performance study demonstrates the low response time of our system while maintaining scalability. In the future, the lightweight CryptoCloud can be smoothly embedded with a browser through an extension application.

KEYWORDS

Cloud storage, data confidentiality, encrypted search, identity-based encryption (IBE), key management, privacy preservation, searchable encryption, secure data sharing.

1 | INTRODUCTION

Cloud computing has reshaped how data is stored and processed by offering elastic, on-demand access to shared, configurable resources over the Internet[1]. Built on advances in virtualization, autonomic and grid computing, and service-oriented architectures, todays cloud platforms support a wide range of consumer and enterprise services across government and industry. The model reduces capital expenditure, simplifies operations, and enables collaboration from any location. Service layers—Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS) provide modular service delivery, while Storage-as-a-Service exposes affordable object storage to end users and applications. A widely cited definition

from NIST describes cloud computing as a pay-per-use model that offers convenient, on-demand network access to shared resources which can be rapidly provisioned and released with minimal management effort[2].

Notwithstanding these benefits, there are serious issues with confidentiality, integrity, lifecycle control, and auditability when data is outsourced to outside cloud service providers[3]. Replicated copies may persist after deletion, and users no longer have direct control over their data or encryption keys. Furthermore, trustworthy but inquisitive suppliers or external adversaries can try to deduce private data from usage patterns or archived content. In reality, there are still significant barriers to broader adoption for sensitive workloads, including insider risks, multi-tenant leakage, compliance and governance constraints, and performance management[4]. Storage needs confidentiality, integrity, access control, and guaranteed erasure, whereas compute needs privacy-preserving and verifiable processing when the cloud performs queries or analytics on users' behalf.

Since it prevents providers from learning plaintext data even if their infrastructure is compromised, client-side encryption is a logical first line of defense (citearif2025comprehensive). While public-key cryptography enables secure distribution and fine-grained control, symmetric-key cryptography offers efficiency with a shared secret. However, there is a usability gap when data is encrypted before upload, since the server-side cannot perform direct keyword searches or selective retrieval after encryption. Security is compromised when decryption keys are shared with the provider, and it is not cost-effective to download and locally decode the entire dataset[6]. The tension between confidentiality and usability requires mechanisms that protect data privacy while enabling efficient server-side search and retrieval.

This problem is addressed by searchable encryption (SE), which enables keyword queries over ciphertexts without disclosing the query terms or the document content to the cloud provider[7]. Before outsourcing documents to the cloud, the data owner in a standard SE system encrypts them along with a searchable index. Only the pertinent ciphertexts are returned by the cloud once authorised users create encrypted query tokens, also known as trapdoors, which are compared to the encrypted index. This research topic includes both symmetric searchable encryption and public-key encryption, each with trade-offs among efficiency, expressiveness, and trust assumptions. Indexing techniques often resemble plaintext retrieval: forward indexes map documents to keywords, whereas inverted indexes map keywords to documents[8]. Because of their efficient sublinear search capabilities, encrypted inverted indexes are commonly used in large-scale systems. Large-scale systems often use encrypted inverted indexes because they offer effective sublinear search capabilities.

Over the last twenty years, searchable encryption has changed dramatically, moving from static datasets to methods that enable regular updates[9] and from single-keyword search to multi-keyword, ranked, and dynamic query models. Minimising information leakage, verifying results, and facilitating multi-user access control have all been investigated in more recent work. Several obstacles still stand in the way of a realistic implementation of commodity cloud storage, notwithstanding these advancements. Many approaches hinder scalability and usefulness in real-world scenarios by requiring additional trusted infrastructure, imposing significant preprocessing or query cost, or depending on intricate cryptographic primitives. It is crucial to consider query unlinkability, as deterministic query tokens can reveal search patterns[10]. Effective multi-user sharing and revocation remain challenging without necessitating data re-encryption or incurring high key-management costs.

Even when data is secured, consumers want cloud storage solutions to deliver rapid search performance, instant access revocation, and easy sharing. Existing searchable encryption solutions, however, frequently prioritise improving discrete security features over comprehensively attending to these system-level needs. In the present cloud storage ecosystems, deployable systems that integrate secure sharing, encrypted search, and usability remain in conflict with theoretically safe searchable encryption technologies.

This study presents a workable system for sharing, storing, and searching encrypted data in untrusted cloud storage. Before being outsourced to the cloud, the suggested method encrypts the documents and the inverted index from beginning to end. Enforcing authorisation, authenticating users, and providing encrypted search terms that render identical queries unlinkable are all handled by a lightweight trusted entity. The cloud provider does not know the content of documents or the search terms being used; it only works with ciphertexts. Only authorised users are granted access to the decryption keys using their own public keys; the supplier never keeps them. It allows for instant revocation without requiring re-encryption of stored data. The main contributions of this work are summarized as follows:

- **Secure multi-user searchable encryption framework:** The proposed framework enables fine-grained keyword search for multiple authorized users while preserving data confidentiality. Search keywords are issued through an authorization mechanism that ensures only legitimate users can perform encrypted queries, maintaining strict access control.
- **Dynamic and privacy-preserving encrypted inverted index:** A secure and dynamic encrypted inverted index is designed to support efficient single- and multi-keyword searches while preventing the leakage of keywords and document contents. The index supports flexible updates and scalable encrypted storage management.

- **Lightweight non-interactive search authorization:** After authorization, encrypted search permissions are generated in a non-interactive manner, supporting dynamic user enrollment without affecting existing users and incurring minimal computation and communication overhead.
- **End-to-end implementation and validation:** A complete web-based framework, CryptoCloud, is implemented in which all cryptographic operations are executed locally, with a stateless backend integrated with existing cloud storage services. Experimental evaluation demonstrates practical performance, scalability, and efficient encrypted search functionality.

The remainder of this paper is organized as follows. Section 2 presents a review of existing research on secure cloud storage and searchable encryption techniques. Section 3 defines the system and threat models, followed by the security requirements in Section 4. Section 5 describes the proposed CryptoCloud framework and its operational design. Section 6 details the system implementation, and Section 7 evaluates its performance. Section 8 provides a security analysis and formal verification results. Finally, Section 9 concludes the paper and outlines future research directions.

2 | LITERATURE REVIEW

Searchable encryption is now a fundamental component of secure cloud storage systems, enabling keyword-based retrieval of encrypted data stored on untrusted servers. Public-Key Encryption with Keyword Search (PEKS) and Searchable Symmetric Encryption (SSE) are the two main cryptographic paradigms that have shaped this field's development over the last 20 years. To increase efficiency and scalability, indexing strategiesin particular, encrypted inverted indexeshave been developed concurrently. The representative works in these areas are reviewed in this section, which also places our framework in this context.

2.1 | Secure Cloud Storage and Early Searchable Encryption

Early studies on safe cloud storage focused more on data sharing, secrecy, and guaranteed destruction than on effective retrieval. To protect data from unreliable providers, Patwary and Palit[11] focused on client-side encryption in their prototype system for safe file sharing and storage in cloud storage infrastructures. Without the need for outside, trusted key managers, their strategy reduced risks such as unauthorized access and the permanence of copied data. However, it was not practicable for large-scale datasets where selective keyword-based retrieval is crucial because, like many early secure storage systems, it required full file download and decryption for access.

The first practical attempt to enable search over encrypted data was introduced by Song *et al.*[12] in 2000, who demonstrated that keyword search could be performed directly on ciphertext without revealing plaintext to the server. Their scheme embeds encrypted keyword information into ciphertext blocks and allows the server to test for keyword presence using a user-generated trapdoor. While this work established the feasibility of searchable encryption and provided strong confidentiality guarantees, it relied on linear scanning of all encrypted documents and did not support indexed search, resulting in poor scalability for cloud environments.

To address efficiency and formal security modeling, Curtmola *et al.*[13] in 2006 introduced Searchable Symmetric Encryption (SSE) with explicit leakage definitions and efficient constructions based on encrypted inverted indexes. Their work clearly distinguished between encryption security and search security by formalizing access-pattern and search-pattern leakage, demonstrating that sublinear and practical search over encrypted data is achievable. This contribution laid the foundation for modern SSE systems but remained focused on private-key settings and did not consider usability, authorization workflows, or real-world cloud integration.

In parallel, Boneh *et al.*[14] proposed Public-Key Encryption with Keyword Search (PEKS) in 2004 to support searchable encryption in open and multi-sender environments. PEKS enables keyword encryption using public keys and search using private-key-generated trapdoors, making it suitable for applications such as encrypted email routing. However, PEKS schemes rely on expensive public-key operations, leak access patterns by default, and are vulnerable to keyword-guessing attacks. Extensions incorporating fine-grained access control using attribute-based encryption, such as the scheme proposed by Chaudhari and Das[15], improve expressiveness but introduce heavy computational overhead and complex trust assumptions.

Islam and Palit[16] built on previous work by introducing a keyword-based search-and-share system for encrypted cloud data that uses an encrypted inverted index and symmetric encryption. That work demonstrated that safe keyword search and restricted sharing were possible in a limited setting. It enabled both single- and multi-keyword searches and reduced unnecessary

downloads, but it failed to address formal threat modelling, system architecture, and scalability. The current work builds on that initial concept by providing a formal security architecture, an improved authorisation and revocation mechanism, and a thorough end-to-end implementation suitable for actual cloud storage settings.

2.2 | Searchable Symmetric Encryption (SSE)

The goals of recent research on searchable symmetric encryption (SSE) have been to facilitate dynamic updates, improve efficiency while maintaining usability in cloud environments, and strengthen security against practical inference attacks. One important area of research examines how adversaries can exploit leaks that occur during encrypted searches and proposes defences against such threats. Li *et al.*[17] examined response identity attacks, in which adversaries can deduce sensitive information about underlying keywords by repeatedly seeing identical query responses. They provide a feasible SSE design that integrates controlled randomisation into search results to lessen the adversary's capacity to correlate related queries while maintaining efficient search performance. Although the system is successful in limiting answer identity leaks, access patterns are still revealed, and long-term query behaviour analysis remains incomplete.

A substantial amount of research also examines query correlation and its impact on SSE security, especially for conjunctive queries. Liu *et al.*[18] showed that correlations between encrypted query tokens can allow for extremely successful query recovery attacks, even when conventional leakages like access and search patterns are reduced. Their work presents a passive approach that produces high keyword recovery rates in practical scenarios and formalises generalised query correlation patterns for conjunctive SSE. Stronger unlinkability methods or query obfuscation techniques are required, as this conclusion emphasises that leakage from user query behaviour itself is a fundamental challenge for SSE.

Several works suggest strong dynamic SSE methods with formal forward and backward security assurances to facilitate dynamic data outsourcing. RO(SE)^2 is a search-efficient and resilient SSE technique that guarantees both forward and backward security while allowing for update failures and inconsistencies, as presented by Yang *et al.*[19]. Resilience is a crucial component of their design as updates can be halted or reorganised in real-world cloud environments. Parallel to this, Bian *et al.*[20] developed dynamic SSE to support conjunctive queries, proposing techniques that make expressive search possible while preserving update security. By implementing a lightweight update search permission control, SEAC[21] further improved this strategy by enabling fine-grained authorisation changes without requiring significant cryptographic overhead. Although these techniques significantly improve practicality, they still reveal some access pattern information and require maintaining additional client-side data.

Research in SSE is still ongoing in the areas of expressive search and leakage reduction. By adding padding and structural modifications to conceal result sizes, SMSSE[22] explicitly addressed size-pattern leakage, which exposes information through the quantity of returned results. A leakage-reduced SSE approach for multi-keyword queries was suggested by Deng *et al.*[23]. This scheme uses carefully crafted index and token structures to minimise information exposure during conjunctive search. A privacy-preserving conjunctive SSE technique, designed for Cloud-IoT healthcare systems, was developed by Ma *et al.*[24] for application-oriented contexts, striking a balance between efficiency and secrecy in a field with stringent privacy regulations. These strategies reduce specific leakage pathways, but they often entail additional storage or processing costs.

Lastly, several studies investigate orthogonal improvements to SSE, such as decentralisation, verifiability, and efficiency optimisation. By optimising index representations and cryptographic operations, Chakraborty *et al.*[25] concentrated on making SSE schemes quicker and smaller while also obtaining significant savings in computation and storage expenses. Although it came at the expense of controlled false positives, BloomSec[26] used Bloom-filter-based indexes to increase scalability and privacy in cloud environments. Verifiable SSE was introduced by Ji *et al.*[27] utilising additive homomorphic encryption to enable users to confirm that search results from an untrusted server are accurate. SEARCHAIN[28] combined SSE with blockchain-based rewarded practical work to offer decentralisation and auditability, but at the expense of higher deployment overhead and system complexity. Together, these studies demonstrate significant advances in SSE research while highlighting ongoing challenges in balancing expressiveness, efficiency, and leakage resilience.

2.3 | Public-Key Encryption with Keyword Search (PEKS)

Public-Key Encryption with Keyword Search (PEKS) has evolved into a rich research area aimed at enabling keyword-based search over encrypted data in open, multi-user environments where shared secret keys are impractical. Unlike symmetric searchable encryption, PEKS allows data senders to encrypt keywords using a receivers public key, enabling any authorized receiver to search encrypted data via trapdoors generated from their private key. While this paradigm naturally supports decentralized data sharing, extensive research has shown that PEKS schemes are inherently vulnerable to keyword guessing attacks (KGA), particularly when the keyword space is small. As a result, much of the PEKS literature focuses on mitigating both outside and insider KGAs through authenticated encryption, designated testers, proxy assistance, and verifiability.

One major research direction strengthens PEKS through authentication and tester restriction. Public-key authenticated encryption with keyword search (PAEKS) requires the senders secret key during keyword encryption, ensuring that even an insider cloud server cannot mount dictionary attacks. However, authenticated PEKS schemes significantly increase computation and trapdoor size in multi-sender settings. To address this, proxy-assisted approaches were introduced, where a trusted proxy converts heterogeneous ciphertexts into a uniform format to enable constant-size trapdoors. More recently, proxy-free authenticated PEKS schemes eliminate reliance on trusted intermediaries by allowing the cloud server to securely update ciphertexts itself, achieving efficient search while resisting insider KGAs *et al.*[29]. Lightweight authenticated PEKS constructions further reduce overhead for multi-user cloud storage, though they still rely on expensive public-key primitives *et al.*[30].

Another active line of work focuses on dynamic, verifiable, and fine-grained PEKS. Dynamic PEKS schemes support document insertion and deletion without rebuilding encrypted indexes, improving applicability to real cloud systems *et al.*[31]. Verifiable PEKS allows users to confirm the correctness of search results returned by an untrusted server, often using homomorphic tags or pairing-based proofs *et al.*[27,32]. However, recent cryptanalytic studies reveal subtle weaknesses in designated-tester PEKS designs: security analyses demonstrate that pairing-based constructions relying on symmetric pairings may invalidate decisional DiffieHellman assumptions, leading to keyword leakage through trapdoor linkability or tester key exposure *et al.*[33]. These results highlight the fragility of PEKS security when strong trust assumptions on servers or testers are relaxed.

Lattice-based PEKS methods have been developed to provide post-quantum security to address long-term security issues. These designs support advanced features such as multi-user authentication, ciphertext updating, and forward security by substituting the Learning With Errors (LWE) assumption for bilinear pairings *et al.*[34]. While lattice-based PEKS significantly strengthens cryptographic guarantees, it incurs substantial ciphertext expansion, complex key management, and high computational cost. Even efficiency-oriented PEKS schemes based on arithmetic span programs or optimized public-key operations remain several orders of magnitude slower than symmetric approaches under high query volumes *et al.*[25]. As a result, PEKS algorithms are unable to adequately meet the latency and scalability requirements of real-world cloud storage systems that often offer multi-keyword searches. These efficiency limitations highlight the fundamental trade-off between robust cryptographic expressiveness and realistic deployment performance. Because of this, symmetric searchable encryption remains preferred in many real-world cloud storage applications that require low latency and high query throughput.

2.4 | Encrypted Inverted Index Techniques

In searchable encryption systems, encrypted inverted indexes are the fundamental data structure that enables effective keyword-based retrieval. Inverted indexes map keywords to document IDs, making them especially appropriate for large datasets with high query volumes, whereas forward indexes link documents to keyword sets. The idea of secure indexes utilising Bloom filters to provide encrypted keyword testing with lower storage overhead was first presented in early work by Goh[35]. Although this method outperformed ciphertext scanning in search performance, it had limitations regarding dynamic updates and false positives. Because of its predictable lookup efficiency and compliance with real-world cloud storage infrastructures, encrypted inverted lists became the predominant indexing approach in later SSE designs.

Inverted index designs were developed to accommodate Boolean and multi-keyword searches as encrypted search capabilities got more powerful. To save search space while maintaining privacy, recent systems use layered or hybrid indexing structures. By integrating classification trees, Bloom filters, and encrypted inverted lists, Fugkeaw *et al.*[36,37] proposed hybrid inverted index frameworks that efficiently enable Boolean searches across encrypted cloud logs and collaborative IIoT data. By eliminating unnecessary ciphertexts early in the search procedure, these methods lower query complexity. However, they add additional information structures that might increase leakage surfaces and rely on auxiliary trust mechanisms like smart contracts or blockchain-based authorisation.

TABLE 1 Comparison of representative searchable encryption schemes most relevant to this work

Scheme	SE Type	Index	Query	Dynamics	Third Party	Key Limitation
Curtmola *et al.* [13]	SSE	Inv-Enc	Single	Static	None	No updates, leakage
Yang *et al.* [19]	DSSE	Inv-Enc	Single	F/B	None	Client state, overhead
Bian *et al.* [20]	DSSE	Inv-Enc	Conjunctive	F/B	None	Query correlation leakage
SMSSE [22]	SSE	Inv-Enc	Multi	Static	None	Padding overhead
BloomSec [26]	SSE	Bloom + Inv	Single	Static	None	False positives
SEARCHAIN [28]	SSE + BC	Inv-Enc	Multi	Static	Blockchain	High latency
Xu *et al.* [34]	PEKS	Inv-Enc	Multi	Dynamic	None	Heavy PK cost
Li *et al.* [29]	PEKS	Inv-Enc	Single	Dynamic	Cloud	Complex updates
CryptoCloud	**SSE-based**	**Inv-Enc**	**Single/Multi**	**Dynamic**	**TTP**	**Trusted TTP**

Another area of research adds decentralisation and verifiability to encrypted inverted indexes. Blockchain-assisted searchable encryption systems allow for tamper detection and result verification by embedding encrypted inverted indexes within distributed ledgers or off-chain storage that is anchored by on-chain commitments *et al.* [38]. These methods improve auditability and transparency, but due to consensus processes and on-chain activities, they come with significant communication costs and delays. Furthermore, it is still tricky to dynamically maintain encrypted inverted indexes in decentralised environments, especially when allowing for high query concurrency, frequent updates, or revocation.

In addition to functionality, new research focuses on measuring and reducing the information leakage that inverted index-based searchable encryption inherently causes. Ahn *et al.* [39] showed that response-size leakage from inverted index queries can allow keyword inference attacks even when access and search patterns are concealed using ORAM (Oblivious RAM). They demonstrated that leakage is a continuum rather than a binary trait by using a quantitative tool to assess the efficacy of volume hiding. Many current systems rely on complex auxiliary infrastructure, heavyweight cryptographic techniques, or additional trust assumptions, even though encrypted inverted indexes remain the most feasible basis for effective searchable encryption in cloud contexts. The need for lightweight inverted index designs that maintain efficiency while carefully controlling leakage and operational complexity is driven by these difficulties.

2.5 | Discussion and Positioning

The aforementioned review demonstrates notable developments in searchable encryption spanning symmetric, public-key, and index-based approaches. Although Searchable Symmetric Encryption (SSE) delivers exceptional efficiency and scalability by using encrypted inverted indexes and dynamic updates, it exposes access patterns, response sizes, and query correlations, leaving the system vulnerable to inference attacks. Techniques like padding, volume hiding, or ORAM-based protection reduce leaks but incur a computational and storage cost, limiting their scalability under high concurrency.

In open environments, Public-Key Encryption with Keyword Search (PEKS) provides stronger cryptographic guarantees and flexible access control. Nevertheless, lattice-based and pairing-based systems entail high computational costs, large ciphertext and trapdoor sizes, and challenging key management. Even optimized or proxy-free variants remain substantially less efficient than symmetric approaches, making them less suitable for frequent encrypted searches in user-centric cloud storage systems.

The most practical basis for searchable encryption remains encrypted inverted indexes, thanks to their sublinear search efficiency and compatibility with commodity cloud storage. However, structural leakage remains a problem, as existing solutions often add layers of verifiability, blockchain integration, or new trust assumptions, thereby increasing system complexity.

In this context, our proposed framework, CryptoCloud, is positioned as a deployable, lightweight alternative. To achieve effective search while preserving controlled leakage under a realistic threat model, it uses an SSE-based encrypted inverted index design. Unlike PEKS-heavy systems, public-key cryptography is used only for secure key distribution, reducing search-time overhead. The framework enables smooth interaction with existing cloud storage providers by avoiding the use of trusted execution environments and decentralised infrastructure, unlike blockchain-assisted techniques. Table 1 shows how CryptoCloud provides a practical compromise for real encrypted cloud storage by balancing usability, security, and efficiency.

TABLE 2 Adversarial Behavior of Each Entity

Entity	Assumption	Behavior Description
TTP	Trusted	Manages secret keys and authorizations. If compromised, it may leak keys and identity mappings, resulting in a complete confidentiality breach.
CSP	Untrusted	Executes protocols correctly but may attempt to infer private information from encrypted data, indexes, or queries.
DU	Semi-trusted	Follows the protocol but may misuse authorized access, replay authorized queries, or collude with other users.
AD	Distrusted	Attempts to eavesdrop, access, or alter data stored in the CSP or accessed by the DU. May observe stored data or data in transit but is limited by TLS/SSL protections.

3 | THREAT MODEL

For the threat model of our proposed framework, we make the following assumptions: the Cloud Service Provider (CSP), the Trusted Third Party (TTP), the Data Owner (DO), the Data User (DU), and an Adversary (AD). Before being stored in the CSP, the DO encrypts the data and its associated indexes. With permission, the DU requests information from the CSP and retrieves encrypted documents. The TTP is responsible for storing the keys, and authorization management. The CSP provides storage and computational resources but is not trusted. It is modeled as an trusted-but-curious adversary that follows prescribed protocols but may attempt to infer sensitive information from encrypted files, indexes, or queries. At no point does the CSP have access to plaintext data. The AD tries to gain access to the documents stored in the CSP, which are not shared with the AD.

Assumptions: Secure by Default

We adopt the following baseline assumptions to reflect practical cloud deployments and bound the adversarys capabilities.

- **Secure Communication:** SSL/TLS is used for all communication between entities to avoid tampering or eavesdropping.
- **Authentication:** The DU's confirmed email address, verified by the TTP, is linked to a folder that the DO shares data access with. The DO, DU, and AD authorise access to the TTP and CSP.
- **Cryptography:** Data and indexes are encrypted using AES, and the distribution of decryption keys is secured using RSA.
- **Key Management:** Strongly random keys are generated, never sent in plaintext, and securely stored.
- **Trusted TTP:** By default, the TTP is considered trustworthy, except for the specific compromise scenario described below.

Adversarial Behavior Summary

In order to make duties and risks more straightforward, Table 2 lists the presumed level of trust for each entity along with the types of misbehaviour they might display. This summary adds value to the threat model by clearly illustrating how those presumptions translate into actual hostile actions.

Under these assumptions, the primary threats arise from inference attempts by the CSP, misuse by authorized users, or compromise of the TTP.

4 | SECURITY REQUIREMENTS

Under the specified threat model, the proposed solution aims to ensure that outsourced data remains safe. The following are the main security requirements:

- **Data Confidentiality:** All outsourced documents must remain secret from unauthorized parties. This includes the CSP, which is presumed to be unreliable, as well as external enemies. The CSP should not be able to retrieve or deduce plaintext material, as it only stores encrypted files and indexes. The TTP verifies that only the DO and authorised DUs can access plaintext data and acquire the required decryption keys.

- **Index Privacy:** Keyword-based enquiries over encrypted documents are made possible by the searchable index, but sensitive metadata must not be revealed. Specifically, the CSP should not be able to infer the index's internal structure, the number of documents associated with each keyword, or the actual keywords. By protecting index privacy, correlations between keywords and documents cannot be inferred by statistical or frequency analysis.
- **Keyword Privacy:** The CSP must not be able to see the search queries issued by the DO or authorised DUs. The CSP should not be able to determine which keyword is being searched for, even when it handles search requests and provides the relevant encrypted results. Patterns that could jeopardise privacy should not be shown by repeatedly searching for the exact phrase. By protecting keyword privacy, the CSP is prevented from creating user-interest or search-habit profiles.
- **Authentication:** Strong authentication is required to ensure that only legitimate entities participate in the system. The TTP is responsible for authenticating both DOs and DUs before issuing search tokens or decryption keys. Similarly, communication among the DO, DU, TTP, and CSP must take place over mutually authenticated secure channels (e.g., TLS/SSL) to prevent impersonation or man-in-the-middle attacks. This guarantees that both entities and communication links are trustworthy.
- **Authorization:** The TTP is responsible for ensuring that DUs explicitly approved by the DO receive access privileges. Unauthorized users must be denied the ability to perform searches or obtain decryption keys. Revoked users lose authorization immediately.
- **Access Control:** Access to encrypted documents and search functionalities must remain under the full control of the DO. Fine-grained policies determine which DUs are permitted to search or retrieve specific data. Even though files are stored on an untrusted CSP, access control ensures that only authorized entities can interact with the data.
- **Revoked Share:** A DU's search and decryption capabilities must be instantly removed from the user if their authorisation is revoked. Reevoked users must not be able to access future data by using previously issued keys, tokens, or query results. If required by policy, forward and backward-secure techniques should be used to guarantee that revocation applies to both recently acquired data and previously outsourced information. This ensures that access control and data confidentiality are upheld even in the presence of revoked users.

5 | PROPOSED FRAMEWORK

CryptoCloud, the proposed framework, creates a secure system for storing, searching, and sharing encrypted data in the cloud. The design facilitates effective retrieval and user-controlled sharing while guaranteeing data confidentiality and query privacy. The basic idea behind CryptoCloud is to encrypt files and their indexes before sending them to the cloud. Through TTP, search queries are converted into encrypted search terms, allowing the cloud service provider to deliver only ciphertexts that match the query without learning the keywords. An outline of the suggested CryptoCloud system architecture is shown in Fig. 1, which also shows how the key entities interact and how their data flows securely. We then outline the scenario-based workflows for Do and DU, providing a summary of the notations used to depict the system clearly.

5.1 | Notation

Before presenting the details of the proposed model, it is useful to introduce the notations used throughout this section. Table 3 summarizes these notations, including files and identifiers, cryptographic keys and key management, encrypted keywords, indexing structures, search and retrieval functions, and sharing operations. These definitions provide a clear reference for understanding the workflows described in the following subsections.

5.2 | Scenario 1: Data Owner Searchable Encryption

This scenario, which focuses on the DO, demonstrates how encrypted data can be outsourced and then securely searched. Scenarios 1(A) and 1(B) for data upload and data retrieval each include two sections. When taken as a whole, these scenarios show how the DO interacts with the CSP and the TTP, ensuring that the CSP uses only ciphertext.

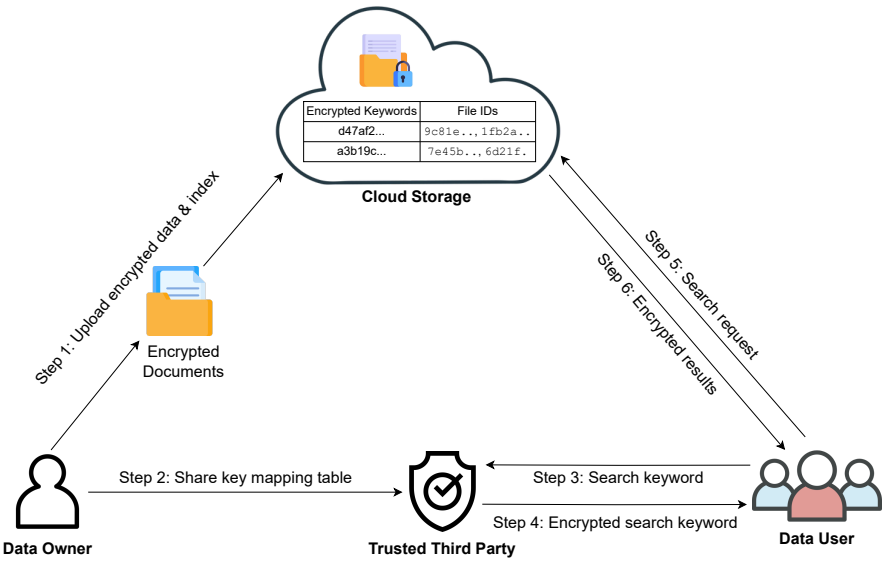


FIGURE 1 System overview of the proposed CryptoCloud searchable encryption framework.

TABLE 3 Notations for Searchable Encryption Scenarios

Symbol / Function	Description	Used In
Files & Identifiers		
f_i	Plaintext file i, $i \in \{1, \ldots, n\}$	All
$\text{Enc}_{rk_i}(f_i)$	Encrypted file ef_i using symmetric key rk_i	All
FID_i	File ID assigned by CSP for encrypted file	All
$FoldID$	Folder ID where encrypted files are stored	2A, 2B
Keys & Key Management		
rk_i	Random symmetric encryption key for file i, generated by DO	All
sk	Secret key for keyword encryption, generated by DO	All
$\mathcal{M}_{rk}$	Key mapping table: $FID_i \leftrightarrow rk_i$, shared with TTP	1, 2A
PK_{DU}	RSA public key generated by DU	2A, 2B
PR_{DU}	RSA private key generated by DU	2A, 2B
$\text{Enc}_{PK_{DU}}(rk_i)$	File key rk_i encrypted under DUs public key	2A, 2B
$\text{Dec}_{PR_{DU}}(\cdot)$	Decrypt data using DUs private key	2A, 2B
Keywords & Encryption		
kw_{ij}	Keyword j for file i (plaintext)	All
$\text{Enc}_{sk}(kw_{ij})$	Encrypted keyword ekw_{ij} for file i	All
kw_s	Search keyword provided by user (plaintext)	All
$\text{Enc}_{sk}(kw_s)$	Encrypted search keyword ekw_s	All
Indexing		
I	Inverted index: keyword $\mapsto \{FID\}$	All
$\text{Enc}_{sk}(I)$	Encrypted inverted index I'	All
Search & Retrieval Functions		
$T(kw_s, sk)$	Encrypted search keyword function: encrypts kw_s to get ekw_s	All
$S(ekw_s, I')$	Search in encrypted index to get F_w	All
F_w	Set of matching File IDs	All
Sharing & Registration		
Share(DU, FoldID)	Share folder with DU (via their registered email)	2A, 2B
RegLink(DU)	Registration link sent to DU for account creation	2B only
Verify(DU)	Verify DU account existence	2A, 2B

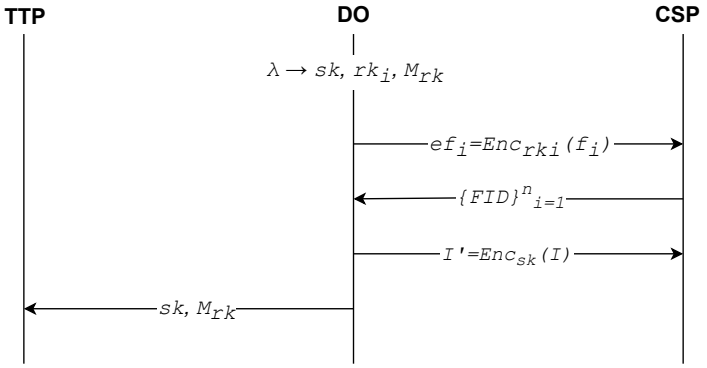


FIGURE 2 Flow diagram of DO data upload.

5.2.1 | Scenario 1(A): Data Upload

Prior to outsourcing, DO produces files and searchable keywords during the data upload phase. Keywords are encrypted with a secret key, and each file is encrypted with a different symmetric key. These encrypted files are then stored at the CSP, which stores only ciphertext and cannot read plaintext data, along with the encrypted index. The files are given distinct identities by CSP, which also stores them and the encrypted index for future searches.DO sends the file key mapping table and the secret key to the TTP to facilitate controlled access and secure search. Only authorised DOs can query and retrieve documents, as the TTP encrypts search terms. Once authorized, DO can issue secure search requests to TTP, while CSP continues to operate only on encrypted data. The overall workflow of this process is illustrated in Fig. 2.

5.2.2 | Scenario 1(B): Data Retrieval

In the data retrieval phase, DO first sends a plaintext keyword and the target folder ID to TTP. TTP encrypts the keyword using the secret key and generates an encrypted search keyword, which it returns to DO. Now, DO searches with the encrypted search keyword in the CSP, which searches the encrypted index for matching entries and returns the identifiers of the relevant encrypted files. CSP only works with encrypted data and never learns the actual keyword or file contents.

Next, DO requests the encrypted files from CSP and sends the corresponding file identifiers to TTP. Using its key mapping table, TTP returns the appropriate decryption keys. Finally, DO decrypts the ciphertexts and recovers the original files. Throughout this process, CSP only handles encrypted data, while TTP ensures that only authorized search and retrieval requests are carried out. The workflow of this phase is depicted in the Fig. 3.

5.3 | Scenario 2: Data User Searchable Encryption

This scenario focuses on the DU and demonstrates how encrypted data shared by the DO can be securely accessed. Scenario 2(A) for registered DU retrieval and Scenario 2(B) for unregistered DU retrieval are the two sections. When combined, these scenarios demonstrate how the DU works with the CSP and TTP to retrieve authorised files and conduct secure searches without disclosing unencrypted information. The ensuing subsections provide the specifics.

5.3.1 | Scenario 2(A): Data Retrieval by Registered Data User

Here, DU searches over the data that DO has supplied. DU and DO first share a folder with several files, and TTP confirms DU's identity and permissions to view the folder. DU sends a folder identifier and a keyword to TTP to start a search, and TTP uses the secret key to create an encrypted search query, then sends it back to DU. After that, CSP receives this encrypted search keyword from DU, verifies the encrypted index, and provides the identifiers of the corresponding encrypted files.

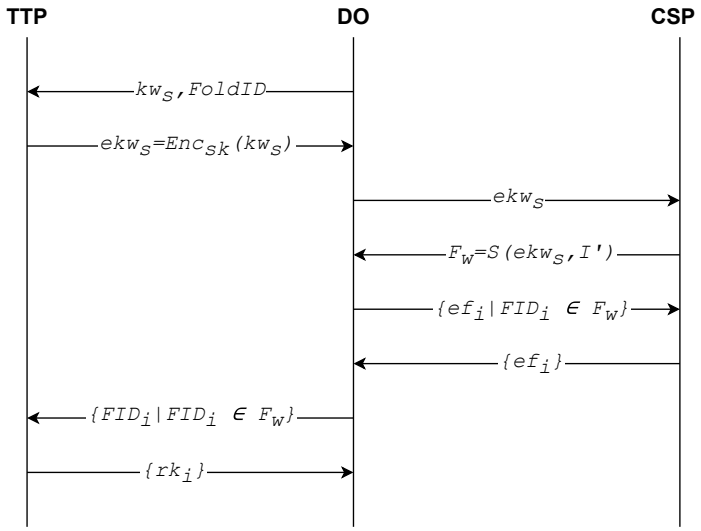


FIGURE 3 Flow diagram of DO data retrieval.

After that, DU obtains the encrypted data from the CSP and requests the matching decryption keys from the TTP. TTP encrypts the file keys using DU's RSA public key before returning them, ensuring secure transmission. DU uses its private key to recover the file keys and then decrypts the files locally. In this way, CSP only sees encrypted data, while TTP ensures authentication and secure key distribution. The flow diagram illustrating this process is shown in Fig. 4.

5.3.2 | Scenario 2(B): Data Retrieval by Unregistered Data User

In this scenario, DU does not register in the system beforehand. When DO shares a folder with DU using an email address, TTP checks whether the DU is already registered. If no account exists, TTP sends a registration link to the DUs email. The DU completes the registration process, after which TTP verifies the DUs identity and authorizes access to the shared folder at CSP as shown in Fig. 5.

Once registration is complete, the retrieval steps are the same as in Scenario 2(A). DU requests an encrypted search keywordfrom TTP, submits it to CSP for a secure search, retrieves the encrypted files, and then obtains the required decryption keys from TTP using an RSA key pair. This guarantees that TTP manages key management and authentication, while CSP only works with encrypted data.

6 | IMPLEMENTATION

This section describes how our proposed concept was implemented as CryptoCloud, a full-stack online application. Using encrypted documents stored in a commodity cloud environment, the system exhibits end-to-end, privacy-preserving keyword search capabilities. With robust security guarantees, the implementation prioritises usability for non-expert users. The technologies, frameworks, and databases utilised are thoroughly described, along with the schema design and guiding concepts.

The two primary elements of the CryptoCloud system are a stateless backend service and a browser-based frontend application. These components use JSON-based RESTful APIs to communicate. To ensure that passphrases, keys, and plaintext data are never revealed to the cloud service provider (CSP) or the backend, all cryptographic operations are performed on the client side. The CSP is Google Drive, and key wrapping, authorisation, and the issuance of encrypted search keywords are managed by a lightweight trusted third-party (TTP) backend. The frontend and backend implementations are covered in detail in the ensuing subsections.

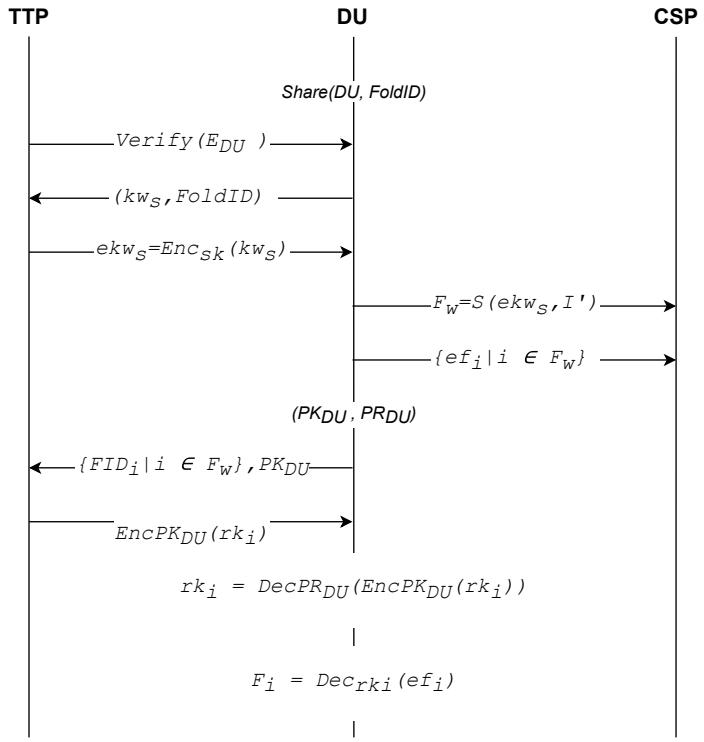


FIGURE 4 Flow diagram of Data Retrieval by Registered Data User.

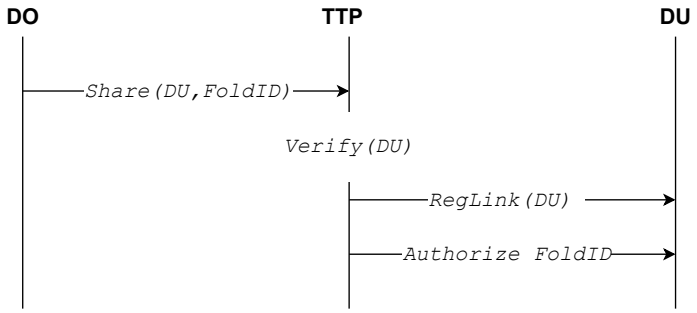


FIGURE 5 Flow diagram of Data Retrieval by Unregistered Data User

6.1 | Dataset Selection

We chose a dataset from electronic health records (EHRs) to assess our deployment. Beyond standard charts, electronic health records (EHRs) include diagnoses, prescriptions, treatment plans, imaging, test results, and allergies. They are patient-centered and real-time, allowing authorised users to safely access various healthcare organisations, including pharmacies, emergency rooms, and labs.

Although cloud-based EHR systems increase efficiency and interoperability, they also pose security risks, particularly through unauthorized access and data breaches. As a result, role-based access control and encryption are crucial for safeguarding private data both in transit and at rest. The EMRBots dataset[40], a database of 100,000 patients that includes 361,760 admissions, 107,535,387 laboratory observations, and 100,000 virtual patients, was used for testing. The dataset is appropriate

FIGURE 6 User sign-up and login interface of CryptoCloud.

for assessing secure searchable encryption because, despite being synthetic, it faithfully captures the volume and complexity of real-world records.

6.2 | Frontend Application

The *CryptoCloud Frontend* component is a cross-platform web application that works flawlessly on both desktop and mobile devices. Complete execution on the user's device guarantees responsiveness and portability. Unlike traditional online applications that could inadvertently disclose sensitive information, such as passphrases or cryptographic keys, to remote servers, the proposed approach enforces client-side cryptographic operations throughout the workflow. In this way, unencrypted data stays in the user's context and is never transferred to outside services.

During development, usability, scalability, and security are carefully balanced by taking into account both internal (architectural) and external (user-facing) factors. Asynchronous data processing guarantees a seamless user experience even during computationally demanding cryptography or I/O operations, while the application's modular architecture improves maintainability and makes future additions easier. Without the need for technical know-how, the user interface's (UI) straightforward design enables the use of secure encryption, search, and sharing functions.

6.2.1 | Functional Specification

This section presents the functional behavior of the CryptoCloud application and its user interface. It describes the main user processes that show how the system works from beginning to end, including important exchanges such as file management, secure search, user registration, authorisation, and sharing.

- **Sign-up and Login:** A user must register for an account with CryptoCloud by entering a strong passphrase and a working email address. In addition to requiring a mix of capital and lowercase letters, numbers, and special characters, the system mandates a minimum length of eight characters. Duplicate registrations or weak passphrases are rejected with the relevant error warnings. Registered users use the same login credentials. Password hashes are securely stored for authentication, and incorrect inputs are reported immediately. As shown in Fig. 6, this process is identical for both DO and DU, ensuring a consistent authentication experience.
- **Cloud Storage Consent:** The CryptoCloud application requires access to a cloud storage service to perform essential functions, including uploading, searching, sharing, and retrieving encrypted documents, after successful registration and login. Because of its extensive use and popularity, Google Drive serves as the cloud storage provider in our solution. The CryptoCloud application never sees user credentials because access is provided via Google's OAuth 2.0 authorization mechanism.

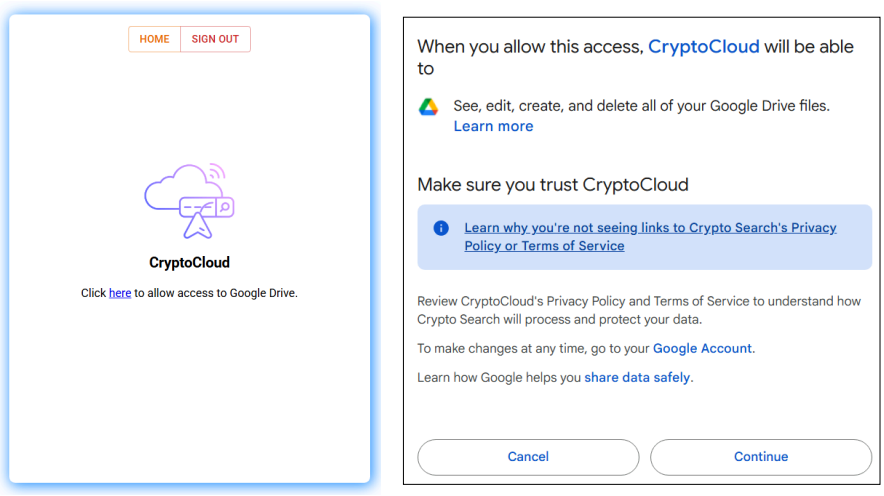


FIGURE 7 Cloud storage access authorization for CryptoCloud.

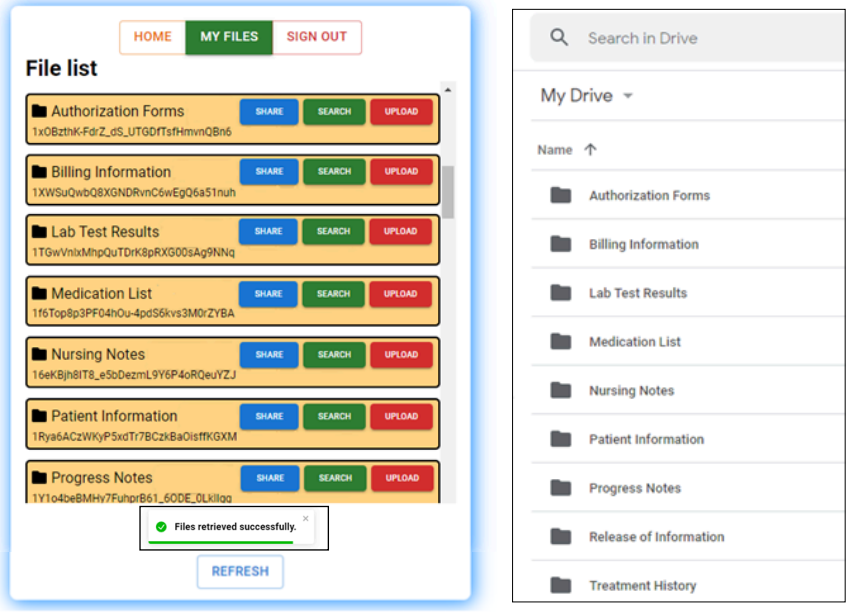


FIGURE 8 Folder management interface of CryptoCloud.

Users allow CryptoCloud to view, create, edit, and remove files from their Google Drive when prompted. This authorisation is only needed once, after which Google's API securely manages access tokens. Crucially, as seen in Fig. 7, storage access is limited to the client-side application; the backend server never handles authentication secrets or unencrypted data.

- **Folder Management:** After login, the DO is presented with a list of folders retrieved from Google Drive and displayed within the CryptoCloud application. Each folder is associated with three core operationsUpload, Share, and Searchenabling secure management of encrypted documents. Folder identifiers are shown beneath each folder name, and a notification panel provides real-time feedback on ongoing tasks. Status messages, such as files retrieved successfully, confirm synchronization between Google Drive and the CryptoCloud application. A refresh option is also provided to retrieve updates, including newly added or deleted files, directly from the cloud, as shown in Fig. 8.

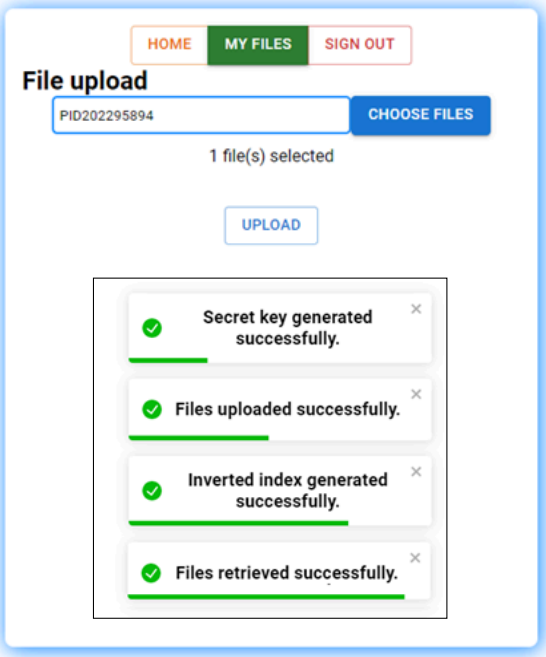


FIGURE 9 Live status notifications: Key generation, file upload, inverted index generation, files retrieve

- **Upload:** The upload workflow enables the DO to add files to a selected folder in Google Drive while preserving end-to-end confidentiality and future searchability. From the folder view, the DO selects Upload, chooses one or more files, and assigns single or multiple keywords. All cryptographic operations are performed locally within the CryptoCloud application: a fresh AES symmetric key is generated for each file, a secret key is generated for keyword encryption, and an encrypted inverted index is constructed. As illustrated in Fig. 9, communication between the CryptoCloud application and the cloud service provider (CSP) occurs over secure SSL/TLS channels, ensuring that only encrypted files and the inverted index are sent.

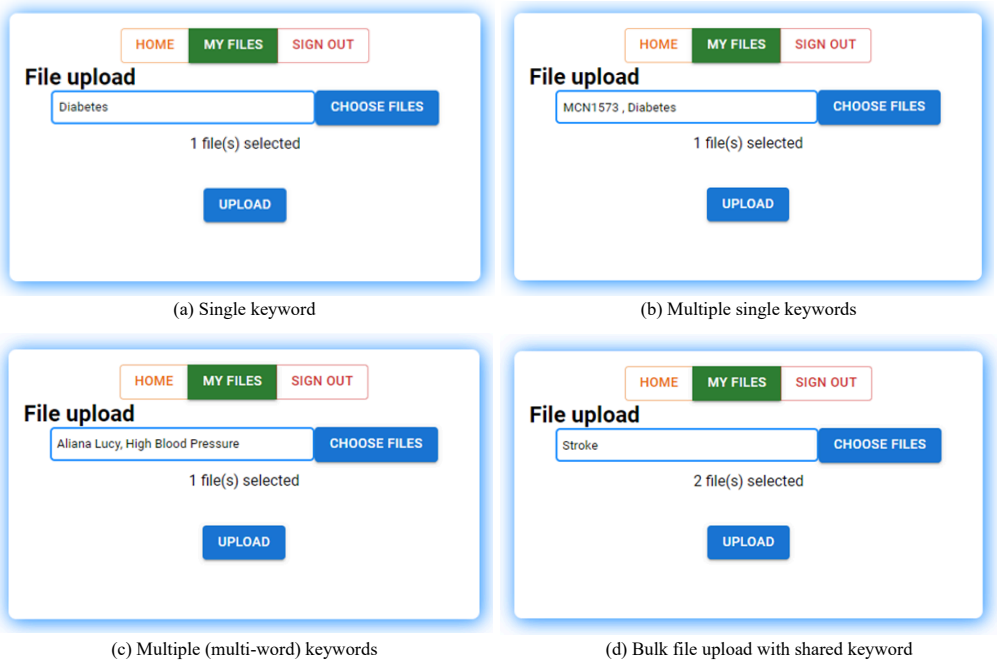
  A notification screen during submission provides the following summary of the upload procedure without disclosing unencrypted data:

  i. **Key generation:** AES symmetric keys are generated for uploaded files.
  ii. **File upload:** Encrypted files are uploaded to Google Drive, and FID is returned for each file.
  iii. **Inverted index update:** Encrypted keywords and corresponding FIDs are inserted into the encrypted inverted index.
  iv. **Folder refresh:** The folder view is refreshed to display newly uploaded files.

  In addition, the DO may assign keywords using four practical modes: single keyword, multiple single keywords, multiple keywords (phrases), and bulk upload with shared keywords. These modes enable efficient encrypted search while keeping the upload process simple for non-expert users, as illustrated in Fig. 10.

  a. **Single keyword:** A single term (e.g., Patient ID, Medicare Number, Medical History, or a disease name such as Diabetes) is associated with a file.
  b. **Multiple single keywords:** Comma-separated terms (e.g., Patient ID, Medicare Number, Medical History) enable discovery of the same file through multiple exact matches.
  c. **Multiple keywords:** Multi-word phrases (e.g., Aliana Lucy, High Blood Pressure) are treated as single keyword units.
  d. **Bulk upload with shared keywords:** Multiple files are uploaded in a single action and associated with a common keyword set (e.g., tagging Patient 4 and Patient 5 records with the keyword Stroke).

  In summary, the upload workflow lets DO adds files securely, attach useful keywords, and see clear status updates as each step completes. All encryption happens in the web interface (in the browser), and only encrypted files and inverted index entries are stored to the CSP. The system supports single files or batches, with single or multiple keywords, so newly added



(a) Single keyword

(b) Multiple single keywords

(c) Multiple (multi-word) keywords

(d) Bulk file upload with shared keyword

FIGURE 10 Upload modes supported in *CryptoCloud*: (a) single keyword; (b) multiple single keywords; (c) multiple keywords; and (d) bulk upload under a shared keyword set.

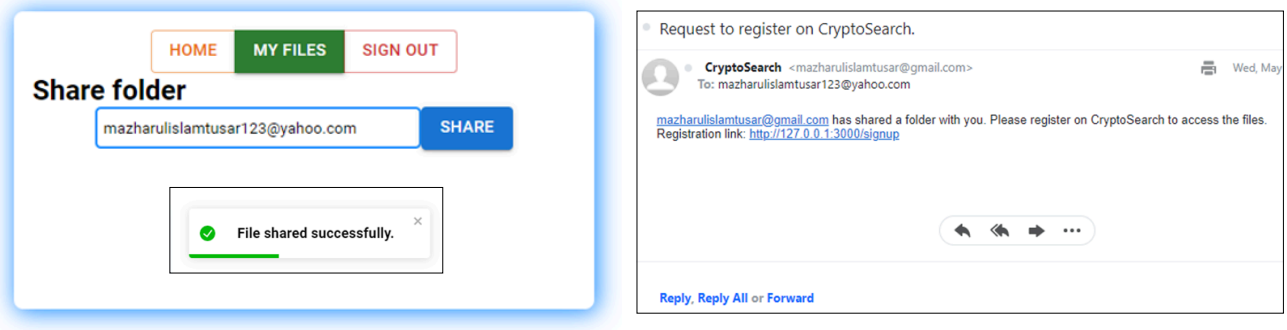
documents become searchable right away. If something goes wrong (for example, a network issue), the interface shows an error and the user can try again without affecting previously uploaded files.

- **Share:** When the DO grants a DU access to a folder, the DU can locate and obtain authorised files. The DO enters the DU's email address and chooses the share option in the folder view to confirm the process. After a successful sharing attempt, a notification (such as "File Shared Successfully") is shown to confirm that the selected recipient may access the folder. The permission structure included in Google Drive is used by the sharing mechanism. Although Google Drive offers several access roles (view, comment, edit), the CryptoCloud application provides a streamlined share action to improve the user experience and maintain a consistent workflow. To enable centralised access control by the DO without disclosing any unencrypted data, folder permissions are created internally for the DU's email address.

  Once the sharing process is complete, the designated user receives an email invitation with a link to sign up for CryptoCloud. If they have not already, the receiver needs to register and log in in order to access the shared content. It guarantees that access will be granted only after a verified account has been linked to the encrypted search terms and decryption keys. The invitation procedure and sharing interface are depicted in Fig. 11, where (a) the folder sharing by email with confirmation is shown, and (b) the invitation email with the registration link is shown.
- **Search:** Both the DO and the DU may use keyword-based queries to find encrypted files because of the search capability, which keeps the plaintext keywords hidden from the CSP. As indicated in Table 4, we utilise five sample EHR papers to demonstrate the workflow, each linked to specified keywords. When a query is submitted by the DO or DU, an encrypted search keyword obtained from the TTP is evaluated by the CSP against the encrypted inverted index. Only the identifiers of matching encrypted files (FIDs) are returned, and the corresponding results are presented to the user with options to download and decrypt the files. The search process supports single-keyword retrieval, multi-keyword retrieval, and retrieval of multiple documents associated with the same keyword, as discussed below.

Single-keyword retrieval

Single-keyword retrieval includes both direct queries and alternative identifiers referring to the same document. In this example, Patient 1 is associated with multiple keywords, including a primary identifier (PID202295894), an alternative identifier (MCN1573), and a medical condition (Diabetes). A search using either the Patient ID or the Medicare Number



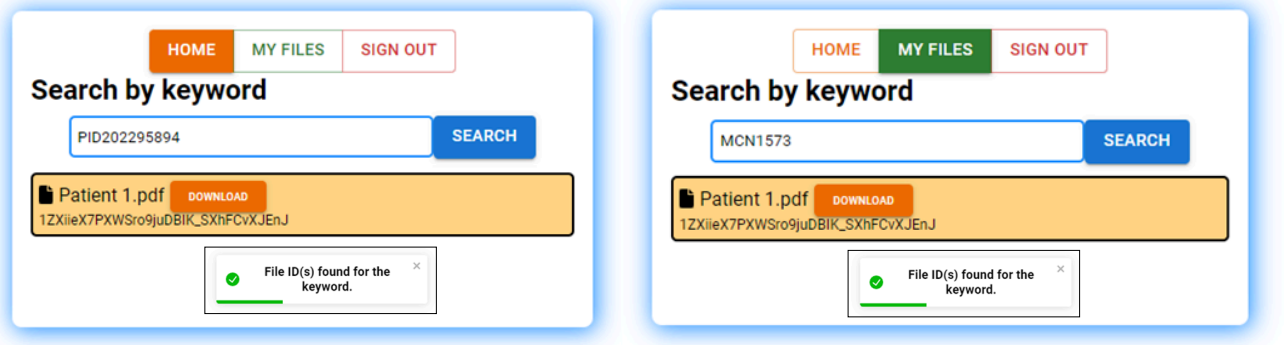
(a) Folder sharing by email with confirmation

(b) Invitation email with registration link

FIGURE 11 CryptoCloud sharing interface: enter DU email and confirm.

TABLE 4 Keywords associated with sample EHR documents used to demonstrate search functionality.

Keywords	FID(s)
PID202295894, MCN1573, Diabetes	Patient 1
Aliana Lucy, High Blood Pressure	Patient 2
Diabetes	Patient 3
Stroke	Patient 4, Patient 5



(a) Query by primary identifier (Patient ID)

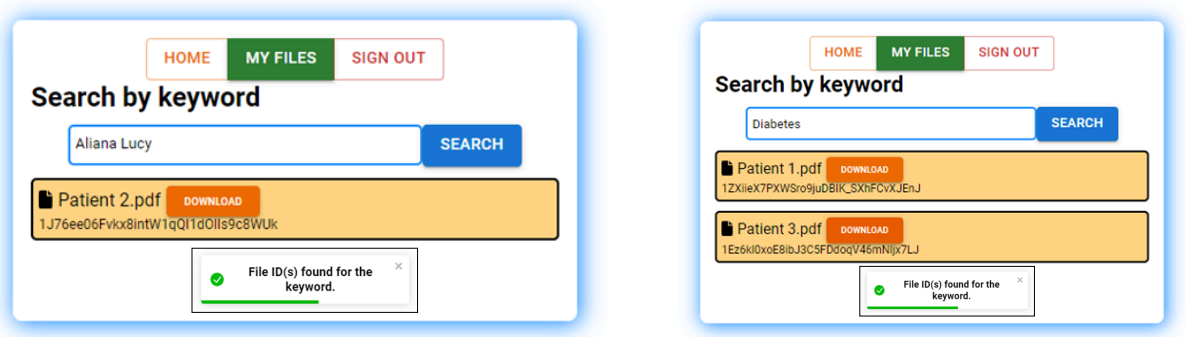
(b) Query by alternative identifier (Medicare Number)

FIGURE 12 Single-keyword retrieval using primary and alternative identifiers.

retrieves the same encrypted file. The single-keyword retrieval process is illustrated in Fig. 12, where (a) shows retrieval using the Patient ID and (b) shows retrieval using the Medicare Number.

Multi-keyword and multi-document retrieval

Additionally, the system allows retrieval of many documents linked to a single keyword and supports keyword-based retrieval utilising multiple query terms. For instance, either Aliana Lucy or High Blood Pressure can be entered to retrieve Patient 2. Similarly, all documents that match a single keyword are returned when it is associated with several files. For example, both Patient 1 and Patient 3 are linked to the keyword Diabetes, yielding several retrieved entries. Fig. 13 provides examples of these situations, where (a) retrieval utilising multiple query phrases and (b) retrieval of several documents associated with the same keyword are demonstrated.



(a) Keyword-based retrieval with multiple queries

(b) Keyword-based retrieval with multiple documents

FIGURE 13 Keyword-based retrieval supporting multiple query terms and multiple matching documents.

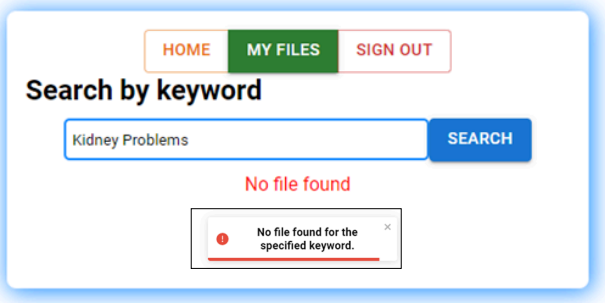


FIGURE 14 Search result indicating no matching files for the queried keyword.

Error handling

When users search for a term that does not match any document in the system, they are clearly informed that no results were discovered. A suitable notification is displayed along with a "No file found" statement on the interface. For instance, Fig. 14 shows that searching for "Kidney Problems" does not produce any results.

6.3 | Backend Application

The proposed system's server-side component, called *CryptoCloud Backend*, functions as a stateless, lightweight API that safely handles metadata and enables controlled access to encrypted documents. The backend handles no encryption keys, passphrases, or plaintext data, as all cryptographic operations are performed entirely on the client side. Instead, it primarily coordinates metadata, manages authorization records, and communicates with the cloud service provider, ensuring that sensitive information remains completely isolated from the server.

The backend architecture emphasizes modularity, maintainability, and scalability. It employs a hybrid design pattern that improves code organization and facilitates future growth by separating request management from business logic. Because the backend and frontend frameworks function independently, there are no dependencies between them, and they can grow and change independently.

6.3.1 | Design Principles

The source code structure and database design were deemed essential for creating the CryptoCloud backend. In the source code, we separated request handling from business logic using a hybrid design pattern. This facilitates future additions, enhances

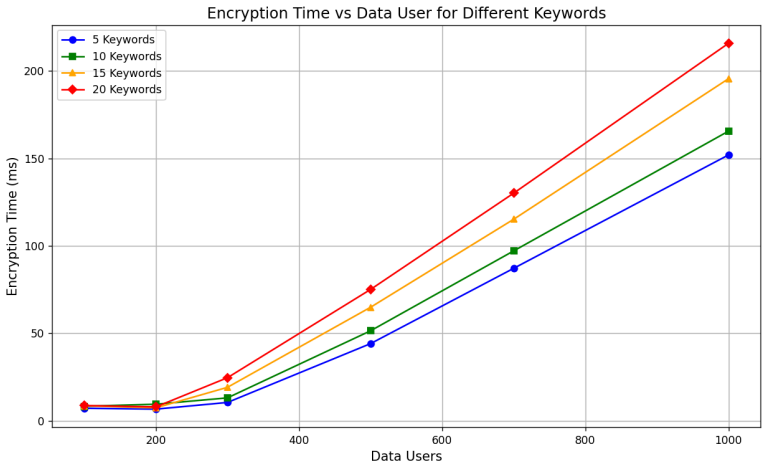


FIGURE 15 Encryption time of Encrypted Search Keywords

organization, and facilitates developer participation. To maintain the separation of concerns, database communication was modularised. To optimise manageability and efficiency, we separated the database into three sections:

- **File Info database** holds properties such as FoldID, FID, the symmetric encryption key, and the file name. Additionally, it keeps track of which DO has permission to access each file.
- **User ID database** The User ID database stores information about DO and DU, including email addresses, hashed passwords, and Google Drive access tokens.
- **Identifiers database** The Identifiers database links user credentials and file metadata, supporting secure retrieval workflows.

This modular design ensures that database operations remain efficient while avoiding performance bottlenecks that could arise from a poorly mixed schema. It also provides a strong foundation for scalability and integration with future system extensions.

7 | PERFORMANCE ANALYSIS

We evaluated CryptoCloud framework under controlled load on a Windows Server 2025 machine with an AMD 64-core CPU and 64 GB RAM. All interactions among DO, DU, CSP and TTP used HTTPS. We evaluate four metrics(a) encryption time for generating Encrypted Search Keywords, (b) search time on cloud storage, (c) search time on local storage, and (d) inverted-index generation timeeach discussed below. The workload is identical across experiments: concurrent users $U \in \{100, 200, 300, 500, 700, 1000\}$ and keywords per user $K \in \{5, 10, 15, 20\}$. Users start simultaneously to emulate a worst-case burst and expose saturation effects.

Encryption time

We isolate the TTPs web service endpoint that transforms a DUs plaintext keyword into an Encrypted Search Keyword and measure the server-side encryption generation time for this operation. In the protocol framework, for each (U, K) pair, a DU submits plaintext keywords and the TTP returns Encrypted Search Keywords. We measure only server-side generation time (no client rendering or CSP I/O). Encryption time grows roughly with K, remains stable at low to moderate U, and shows queuing-driven increases only at higher concurrency. The encryption time is shown in Fig. 15.

Search time on cloud storage

Next, given an Encrypted Search Keyword, the CSP evaluates it over the encrypted inverted index and returns matching ciphertext identifiers. Latency increases with both U and K, with sensitivity to U more pronounced as K grows. The cloud search time is shown in Fig. 16.

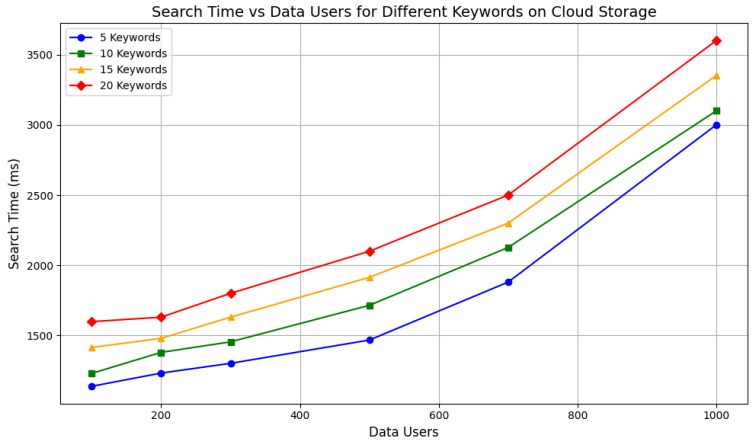


FIGURE 16 Cloud search time

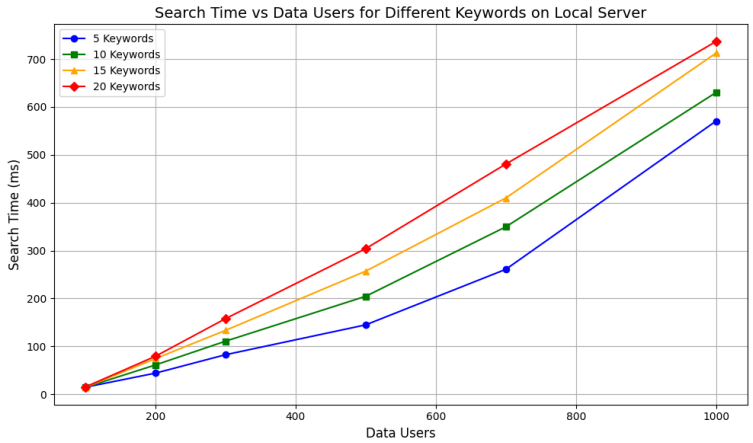


FIGURE 17 Local search time

Search time on local storage

To factor out CSP and network effects, we repeat the experiments on local storage. Trends with respect to U and K are similar, but absolute latencies are consistently lower than on the cloud. The local search time is shown in Fig. 17.

Comparing the search time of cloud and local storage, the gap quantifies CSP and network overheads rather than cryptographic or indexing costs. The gap is modest at low to moderate concurrency and widens at higher U and K. Nearly parallel slopes indicate predictable scaling of the encrypted lookup.

Inverted-index generation time

We measure inverted-index generation as the number of documents grows for different keyword counts. Build time increases with the total number of keyword–document pairs produced by (U, K). Because index creation is performed at data-preparation time and supports incremental updates, this cost is amortized across subsequent searches. The inverted-index generation time is shown in Fig. 18.

Across all experiments, encrypted search keyword generation is lightweight and stable under moderate concurrency; end-to-end search latency on cloud storage is primarily influenced by CSP/network overhead relative to local storage; and inverted-index generation scales with documents and keywords. Overall, CryptoCloud maintains practical responsiveness under realistic multi-user loads while preserving end-to-end confidentiality.

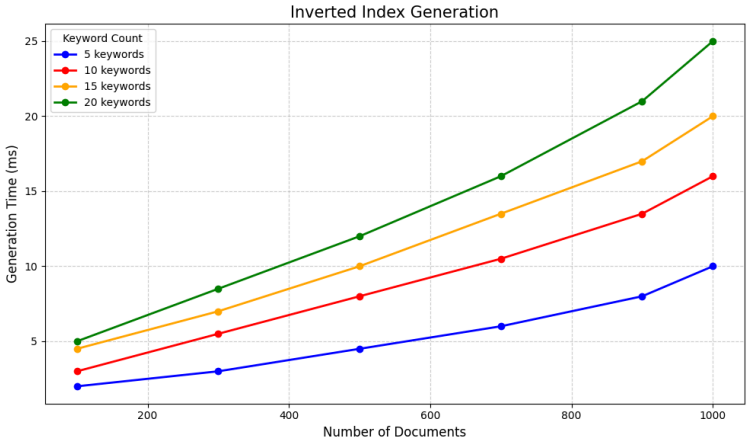


FIGURE 18 Inverted-index generation time

8 | SECURITY ANALYSIS

This section presents a security analysis of the proposed framework and demonstrates that it satisfies the security requirements defined earlier. We assume an untrusted CSP; a minimally TTP that authenticates DO and DU, issues deterministic encrypted search keywords, and holds the encryption keys; and a backend database server that stores user passphrases for authentication only. Documents and per-file indexes are encrypted prior to outsourcing to the CSP, and all inter-component communications occur over secure channels. Below, we show how the design meets each stated requirement.

- **Data confidentiality:** All documents are encrypted on the client side before upload, and the index is stored only in encrypted form. Encryption keys are maintained by the TTP; passphrases are kept at the web application database server for authentication. The CSP therefore observes only ciphertexts and cannot infer document content. Only the DO and authorized DUafter authorizationreceive the wrapped keys required to decrypt locally.
- **Index privacy:** The DO selects plaintext keywords for each file, encrypts those keywords, and uploads the resulting encrypted keywords to the CSP as part of the inverted index that maps encrypted keywords to encrypted file identifiers. Because the CSP sees only encrypted keywords and encrypted identifiers, it cannot read or infer the underlying keywords or document contents from the index.
- **Keyword privacy:** Queries are issued using deterministic encrypted search keywords generated by the TTP. For a given encrypted keyword, the encrypted search keyword is stable across sessions; repeated searches therefore produce the same token. While this allows the CSP to test token equality, it still cannot recover the underlying keyword or any document content and learns only whether the token matches entries in the encrypted index.
- **Authentication:** The framework enforces DO-defined access control at the TTP. The DO specifies which DUs may search or decrypt specific folders/documents; the TTP checks this policy before servicing any request. Only DUs explicitly approved by the DO receive authorization artifacts: the TTP issues deterministic encrypted search keywords and releases wrapped decryption keys only when the DOs policy permits it.
- **Access control:** The DO retains full control over who may search and decrypt. The CSP returns only encrypted documents. Performing a search requires a valid encrypted search keyword, and decryption requires the corresponding wrapped file keyi.e., the per-document symmetric key encrypted under the DUs public key and issued by the TTP. Consequently, even if a DU downloads an encrypted file, they cannot recover plaintext without unwrapping this file key with their private key. Fine-grained, DO-defined policies at the TTP govern both encrypted search keyword issuance and wrapped-key release.
- **Revocation:** When a DU is revoked, the TTP immediately stops issuing encrypted search keywords and wrapped file keys to that user. Because encrypted search keywords are obtained from the TTP at query time, a revoked DU cannot perform new searches; and without new wrapped file keys, the user cannot decrypt subsequently retrieved documents.

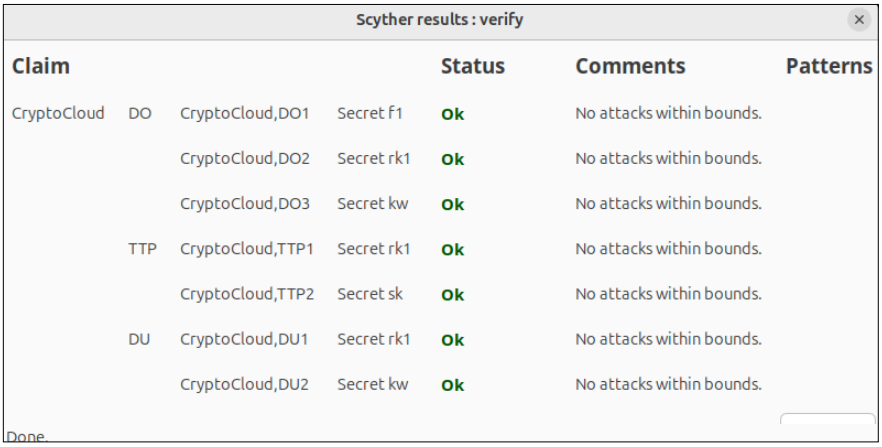


FIGURE 19 Scyther verification results showing satisfied confidentiality and authorization claims.

Taken together, encrypted-at-rest storage on the CSP, encrypted keyword indexing, DU-initiated encrypted search keyword-based search using TTP-issued tokens, authenticated and authorization-scoped release of wrapped file keys, and immediate revocation provide end-to-end protection under the stated threat model and demonstrate that the framework achieves its intended security guarantees.

8.1 | Formal Verification using Scyther

To formally validate the correctness and confidentiality of the message exchanges in the proposed CryptoCloud framework, we performed protocol verification using Scyther[41]. Scyther is an automated security protocol analysis tool widely used in applied cryptography and protocol verification. It operates under the Dolev–Yao adversarial model, in which the adversary has full control over the communication network and may intercept, replay, or modify messages, but cannot break cryptographic primitives.

The modeled protocol captures the end-to-end interaction among DO, TTP, CSP, and DU during encrypted keyword search and controlled file decryption. Two fundamental security attributes that complement our system goals are the focus of the verification:

- **Confidentiality:** Secrecy of plaintext search terms, per-file encryption keys, and keyword encryption keys
- **Authorization:** Guarantee that the decryption keys for retrieved ciphertexts can only be obtained by authorised and authenticated users.

Scyther reported no attacks for all specified secrecy and authorisation assertions after analysing the protocol under unbounded session executions. The study verifies explicitly the confidentiality of: (i) the symmetric encryption keys (rk) that are utilised for every file; (ii) the encryption key for keywords (sk); and, lastly, (iii) the plaintext query keywords (kw). Also, it confirms that only authorized DUs release decryption keys, in compliance with the access-control policy imposed by the TTP. These results provide a formal confirmation of the security properties discussed earlier in this section.

The sequence of security-relevant message exchanges involved in encrypted keyword search and controlled key distribution is highlighted in the protocol message-flow diagram created by Scyther, which illustrates the interactions between DO, DU, TTP, and CSP in Fig. 19.

The Scyther verification output demonstrates that the CryptoCloud protocol is resistant to man-in-the-middle, replay, and key-extraction attacks under the assumed threat model by confirming that all confidentiality and authorisation claims are satisfied without violations, as illustrated in Fig. 20.

9 | CONCLUSIONS

This work proposed a secure framework of storage, sharing, and searching based on searchable encryption to address challenges in confidentiality and usability within cloud environments. The web application we implemented and deployed integrates cryptographic techniques to protect user data while enabling efficient keyword search and controlled sharing of encrypted documents. The use of an inverted index and a trusted third-party entity for key management and encrypted search keyword generation allows this framework to support end-to-end secure workflows: user registration, key generation, encryption, uploading, searching, downloading, and sharing. The model emphasizes modularity, efficiency, and user-friendliness while remaining compatible with existing cloud storage providers without requiring any modifications. This prototype shows that such frameworks can be used in real-world situations, but only in areas where strict privacy obligations apply to sensitive organisational or personal data, such as financial or medical information. Experimental analysis further demonstrates that the proposed scheme strikes a practical balance among memory overhead, search efficiency, and security, thereby improving upon existing approaches.

However, there are still many weaknesses. Reliance on a trusted third party may introduce single points of failure. The mandatory sign-up process can add an extra layer of complexity to native cloud sharing. The performance is also slowed down by computationally expensive cryptographic operations, especially those involving RSA. Another weakness of this framework is that the search is restricted to exact keyword matches. The directions for future work include removing the dependency upon the trusted third party, refining the indexing strategies using tree or graph structures or vectors, and extending search to accommodate a richer query model. Other extensions, such as shareability without registration, lightweight cryptography to reduce computational overhead, and user-side task management, are future aspects that offer hope for developing the framework into a more secure, scalable, and user-friendly facility for privacy-preserving cloud storage and retrieval.

REFERENCES

1. Mathur P. Cloud computing infrastructure, platforms, and software for scientific research. *High Performance Computing in Biomimetics: Modeling, Architecture and Applications.* 2024:89–127.
2. Mell P, Grance T, others . The NIST definition of cloud computing. 2011.
3. Ma Q, Qin Y, Zhu C, Gao L, Chen X. Joint resource trading and task scheduling in edge-cloud computing networks. *IEEE Transactions on Networking.* 2025.
4. Chippagiri S. A Study of Cloud Security Frameworks for Safeguarding Multi-Tenant Cloud Architectures. *International Journal of Computer Applications.* 2025;975:8887.
5. Arif T, Jo B, Park JH. A Comprehensive Survey of Privacy-Enhancing and Trust-Centric Cloud-Native Security Techniques Against Cyber Threats. *Sensors.* 2025;25(8):2350.
6. Jin H, Luo Y, Li P, Mathew J. A review of secure and privacy-preserving medical data sharing. *IEEE access.* 2019;7:61656–61669.
7. Andola N, Gahlot R, Yadav VK, Venkatesan S, Verma S. Searchable encryption on the cloud: a survey. *The Journal of Supercomputing.* 2022;78(7):9952–9984.
8. Chandwani G, Ahlawat A, Dubey G. An approach for document retrieval using cluster-based inverted indexing. *Journal of Information Science.* 2023;49(3):726–739.
9. Sharma D. Searchable encryption: A survey. *Information Security Journal: A Global Perspective.* 2023;32(2):76–119.
10. Cai C, Weng J, Yuan X, Wang C. Enabling reliable keyword search in encrypted decentralized storage with fairness. *IEEE Transactions on Dependable and Secure Computing.* 2018;18(1):131–144.
11. Patwary MSA, Palit R. A Prototype of a Secured File Storing and Sharing System for Cloud Storage Infrastructure. In: IEEE. 2019:1–8.
12. Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: IEEE. 2000:44–55.
13. Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In: 2006:79–88.
14. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public key encryption with keyword search. In: Springer. 2004:506–522.
15. Chaudhari P, Das ML. Privacy preserving searchable encryption with fine-grained access control. *IEEE Transactions on Cloud Computing.* 2019;9(2):753–762.
16. Islam M, Palit R. A Keyword Based Searching and Sharing Scheme on the Encrypted Cloud Data. In: IEEE. 2023:1–6.
17. Li S, Jing X, Wang Y, Xu X, Zhang Z, Wang J. Practical searchable encryption scheme against response identity attacks. *Information Sciences.* 2025;706:121975.
18. Liu H, Xu L, Liu X, Mei L, Xu C. Query Correlation Attack Against Searchable Symmetric Encryption With Supporting for Conjunctive Queries. *IEEE Transactions on Information Forensics and Security.* 2025.
19. Yang X, Wang Q, Qi S, Li K, Qi Y. RO (SE) 2: Search-Efficient Robust Searchable Encryption with Forward and Backward Security. *IEEE Transactions on Computers.* 2025.
20. Bian B, Xu Q, Liu J, Wang J. Practical Robust Dynamic Searchable Symmetric Encryption Supporting Conjunctive Queries. In: Springer. 2025:63–83.
21. Hu Z, Wang J, Chen Z, et al. SEAC: dynamic searchable symmetric encryption with lightweight update-search permission control. *Cybersecurity.* 2025;8(1):75.
22. Yang Y, Fan H, Zhang J, Li B, Ma H, Gu X. SMSSE: Size-pattern mitigation searchable symmetric encryption. *IEEE Transactions on Information Forensics and Security.* 2025.

23. Deng Q, Chen L, Zhu Y, Mu Y. Leakage Reduced Searchable Symmetric Encryption for Multi-keyword Queries. *IEEE Transactions on Cloud Computing.* 2025.
24. Ma J, Peng T, Bei G, Waqas M, Alasmary H, Chen S. Efficient privacy-preserving conjunctive searchable encryption for Cloud-IoT healthcare systems. *ACM Transactions on Privacy and Security.* 2025;29(1):1–27.
25. Chakraborty D, Majumder A, Samajder S. Making searchable symmetric encryption schemes smaller and faster. *International Journal of Information Security.* 2025;24(1):10.
26. Khan AN, Naveed A, Mehmood A, Arora D, Khan AUR, Ali J. BloomSec: Scalable and privacy-preserving searchable encryption for cloud environments. *PLoS One.* 2025;20(12):e0336944.
27. Ji L, Li J, Zhang Y, Lu Y. Verifiable searchable symmetric encryption over additive homomorphism. *IEEE Transactions on Information Forensics and Security.* 2025.
28. Zhao J, Yu J, Yuan X, Liu JK, Zuo C, Cui H. SEARCHAIN: Searchable Encryption As Rewarded-Useful-Work on Blockchain. In: Springer. 2025:368–384.
29. Li H, Susilo W, Shen J, Wang C, Cheng L, Huang Q. Proxy-free Public-key Authenticated Updatable and Searchable Encryption for Cloud Storage. *IEEE Transactions on Dependable and Secure Computing.* 2025.
30. Xu Y, Cheng H, Li J, Liu X, Zhang X, Wang M. Lightweight Multi-User Public-Key Authenticated Encryption With Keyword Search. *IEEE Transactions on Information Forensics and Security.* 2025.
31. Han M, Xu P, Susilo W, Wang W. Dynamic searchable public-key encryption and its application. *Frontiers of Computer Science.* 2026;20(4):2004802.
32. Wang P, Wang H, Ouyang F, Yang Y. Secure Fine-Grained Encrypted Keyword Search for Internet of Things. In: 2025:93–97.
33. Hayashi M, Emura K. Security Analysis on a Public-Key Inverted-Index Keyword Search Scheme with Designated Tester. *Cryptology ePrint Archive.* 2025.
34. Xu S, Chen X, Guo Y, et al. Lattice-based forward secure multi-user authenticated searchable encryption for cloud storage systems. *IEEE Transactions on Computers.* 2025.
35. Goh EJ. Secure indexes. *Cryptology ePrint Archive.* 2003.
36. Fugkeaw S, Deevijit J, Ueasathitwong P, Thanyasukpaisal T. EVSEB: Efficient and Verifiable Searchable Encryption with Boolean Search for Encrypted Cloud Logs. *IEEE Access.* 2025.
37. Fugkeaw S, Deevijit J. Se-collab: Achieving fine-grained and efficiently verifiable searchable encryption with boolean multi-keyword search for collaborative iiot data sharing. *IEEE Access.* 2025.
38. Gao H, Zhang Y, Wang L, Liu Z, Ma Z. A searchable encryption scheme for blockchain-based digital twins. *IEEE Internet of Things Journal.* 2025.
39. Ahn K, Kwak C, Koo D, Hahn C, Hur J. Quantifying Security in Volume-Hiding Searchable Symmetric Encryption Schemes With a Novel Scoring Metric. *IEEE Access.* 2025.
40. Kartoun U. EMRBots: a 100,000-patient database. figshare, Dataset; 2018
41. Cremers CJ. The scyther tool: Verification, falsification, and analysis of security protocols: Tool paper. In: Springer. 2008:414–418.

AUTHOR BIOGRAPHY

MD. MAZHARUL ISLAM received the B.Sc. and M.Sc. degrees in computer science and engineering from North South University, Dhaka, Bangladesh, in 2018 and 2020, respectively. He has served as a Research Assistant with the Institute for Advanced Research (IAR) Lab (United International University in collaboration with North South University) and with the Cyber-Physical System Research Lab at North South University. He received the North South University Vice-Chancellor's Gold Medal in 2024. His research interests include cybersecurity, artificial intelligence, machine learning, searchable encryption, privacy-preserving systems, blockchain for healthcare, and cyber-physical systems. He has published papers in several prospective conferences and journals.

MD. SHAHADUL ALAM PATWARY received his B.Sc. and M.S. degrees in computer science and engineering from North South University, Bangladesh, in 2017 and 2018, respectively. He is currently working as a Senior Software Engineer at Cefalo Bangladesh Ltd., a Norwegian software company. His professional expertise includes high-performance system design and software architecture, with a strong

focus on building scalable, distributed, and resilient computing platforms. His research and technical interests include distributed systems, performance engineering, concurrency and parallel computing, cryptography and secure systems, and large-scale system architecture.

Salekul Islam (Senior Member, IEEE) has been a Professor in the Electrical and Computer Engineering Department of North South University since 2024. Previously, he served as a Professor in the Computer Science and Engineering Department of United International University (UIU) as well as the Head of the Department. He has also held positions as Director of the Institutional Quality Assurance Cell (IQAC) and the Centre for AI and Robotics (CAIR) at UIU. He completed his Ph.D. in computer science from Concordia University, Canada, in 2008, and worked as an FRQNT Postdoctoral Fellow at INRS, Canada, from 2008 to 2011. His research areas include NLP, image processing, cybersecurity, blockchain, edge cloud computing, robotics, and engineering education. He has served as a Senior Editor of *IEEE Access* and as an Associate Editor of *Frontiers in High Performance Computing*.

RAJESH PALIT (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2011. He is currently a Professor with the Department of Electrical and Computer Engineering, North South University, Bangladesh. He has published more than 50 research papers in international conferences and journals. His research interests include cloud computing and distributed systems, data networking and information security, and ICT for development.

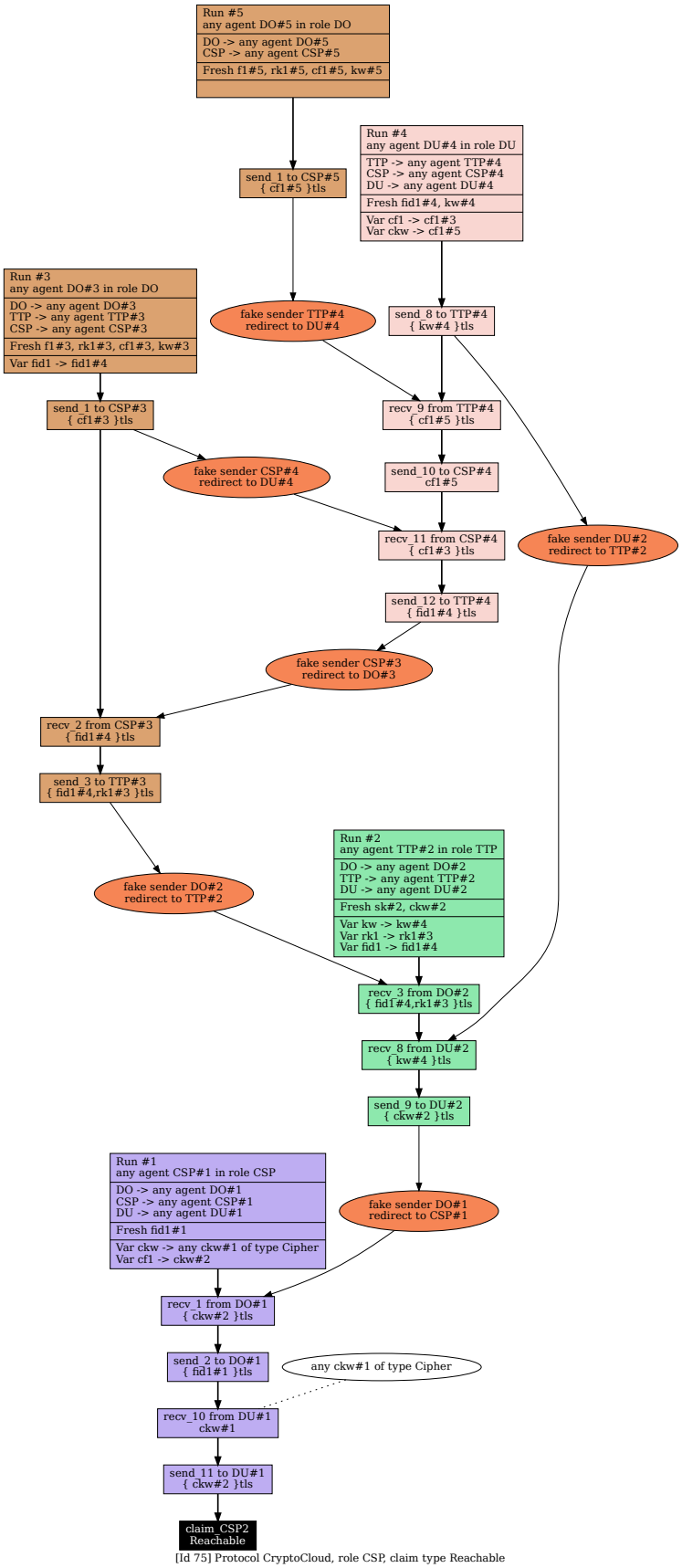


FIGURE 20 Protocol message-flow diagram of the CryptoCloud framework generated by Scyther.