A Keyword Based Searching and Sharing Scheme on the Encrypted Cloud Data

Mazharul Islam
Electrical and Computer Engineering
North South University, Dhaka
mazharul.islam1@northsouth.edu

Rajesh Palit
Electrical and Computer Engineering
North South University, Dhaka
rajesh.palit@northsouth.edu

***Abstract*—Security is one of the major concerns in the cloud server for users when local storage data is outsourced to the cloud. It gives rise to security concerns involving data confidentiality even after the deletion of data from cloud storage. Data must be encrypted before uploading to the public cloud to ensure data security and protection from unwanted and unauthorized parties so they cannot get access to private data. Sharing encrypted data with other users, specifically with a searching facility in the encrypted data, is challenging. Searching over encrypted data is essential to avoid downloading large data to the local storage and searching in the downloaded data. A cloud storage service can search over encrypted data on behalf of the data users using searchable encryption without knowing the underlying plain texts. While many searchable encryption schemes show provable security, they usually expose some query information, making them less practical. This paper presents a noble scheme that enables sharing and keyword-based searching on encrypted data stored in the cloud storage. It helps users download specific data by searching keywords and avoid downloading all the data from the storage.**

***Index Terms*—cloud storage, encrypted data, keyword search, assured deletion**

I. Introduction

Cloud computing provides universal and on-demand access to flexible processing and storage resources. It provides high data availability, simple access to data, and reduces infrastructure costs by outsourcing data to remote servers [1]. Cloud computing enables an economic paradigm where more sensitive data, such as personal health records, emails, government documents, and confidential company documents, is outsourced for data storage.

Although data outsourcing lowers customers' computing and storage expenses, protecting the privacy of sensitive data remains a serious problem [2]. It is difficult for a data owner to authorize their private data to suspicious third-party Cloud storage. This problem arises because the user can no longer physically control the data once transferred to a cloud server. Cloud storage could be curious about sensitive data. Thus, securing sensitive data stored on the cloud is necessary. The most often used technique for protecting user privacy is encrypting user data before outsourcing to the cloud. However, the default search features of the cloud service can't help data users to search in the user-encrypted data. The method of searching and downloading specific data presents a significant challenge nowadays.

Searching is one of the most valuable and fundamental operations many users wish to carry on the data stored on a remote server [3]. The user can search by query and get the expected data from the cloud storage. However, retrieving encrypted data from cloud storage isn't easy. The simplest option is to download all encrypted data to the local device, then run a decryption query. However, this technique demands much communication and computing resources and wastes a lot of bandwidth and computing power.

The existing system models of encrypted search have many limitations and complex functionalities, making the models more complicated. In the literature, some schemes support only a single keyword, and others support only multiple keyword searching, whereas the user may want to search single and multiple keywords. In some models, adding a keyword to an index is easier, whereas deleting a keyword is more complicated. Most of these schemes let the cloud server search for encrypted keywords where the cloud server learns the searching pattern. The proposed scheme is relatively simple and overcomes the above limitations.

It is proposed a solution of searchable encryption where the user can search encrypted documents from a folder where many documents are stored in the cloud storage. It is used inverted index functionality to link each keyword with the related documents. The inverted index contains the keyword and file ID of the encrypted documents. The user must search on the inverted index by keyword for a particular document. If the keyword gets matched, it will provide the file ID of the related document. The user can get that document from cloud storage by file ID. It is a scheme to obtain accurate search results quickly for users and protect sensitive documents from cloud storage and unauthorized entities. The contributions are summarized in the following.

- **Multi Data users:** The scheme provides multiple data users based on identity-based encryption to search for encrypted data that the data owner shares.
- **Search Efficiency:** It is allowed both single and multi-keyword searches and provides matched data retrieval instead of returning irrelevant results.
- **Inverted Index Structure:** The scheme is built based on an inverted index structure that achieves greater flexibility and efficiency, which is dynamic and easy to update.

The rest of the paper is organized as follows. Section II discusses literature reviews on related work on encrypted searching. The discussion on how the proposed model works are given in section III. Limitations of the proposed scheme and related future work are in Section IV.

II. Literature Review

Song *et al.* first proposed a searchable symmetric encryption (SSE) scheme [4] where the authors used an encrypted index for searching. The technique requires the server to go through each document word by word; hence the search complexity increases linearly with the document size. The server needs to scan the entire set of documents. In their model, the authors did not consider a cloud storage system. Searchable encryption schemes have been studied by many researchers afterward.

Patwary and Palit designed a system using the existing cryptography approaches, ensuring the deletion of files by securing the file storing and sharing system for cloud storage servers [5]. The primary focus of their model is to ensure user privacy and security through a user-friendly system that is less computationally intensive and does not rely upon any trusted entity. Their prototype does not have a provision for document sharing and data integrity checking. If data gets changed in cloud storage, there is no way to know if the data was modified in the cloud storage.

In another searchable encryption system proposed by Chaudhari and Das [6], a data owner allows a group of data users to search cloud storage files with fine-grained access control. The cloud storage confirms the user's access privileges before returning the search results. This flexible search control further protects the downloaded files' privacy. Their proposed scheme allows a single keyword search at a time. The major disadvantage of this approach is that it initiates separate searches for each keyword when a data owner or user conducts a file search for many keywords simultaneously, which incurs communication and computation overhead.

A secure inverted index-based search was first introduced by Curtmola *et al.* [7]. Using a uniquely designed linked list data structure, they presented an indexing scheme with the highest time-efficient search function. However, the inverted list's position and content are exposed to the cloud storage server once a keyword is searched. Moreover, updating an existing keyword in the inverted index construction scheme is complicated.

Wang *et al.* [8] proposed an inverted index of a single keyword with strong privacy in which the scheme eliminates the one-time-only search restriction. The design has no probabilistic trapdoor-generating algorithm, making it vulnerable to the search pattern. Buecenna *et al.* [9] proposed a secured inverted index with access control management using two key techniques, homomorphic encryption and the dummy documents technique. Their system performs poorly due to dual inverted index computation and large memory overhead.

Goh [9] defined secure indexes and formulated security schemes to achieve index security and then introduced a secure index construction scheme called Z-IDX based on bloom filters and pseudo-random functions. Bloom filters are memory-efficient data structures, and pseudo-random functions are deterministic and efficient functions that produce pseudo-random values indistinguishable from random sequences. Z-IDX was tested to implement searches over encrypted data. It is an efficient method when considering index updates; however not efficient in search time at the cloud server.

Kamara and Papamanthou [10] proposed a new search scheme based on a tree-based index, which can handle dynamic updates on document data stored in leaf nodes. However, their scheme is designed only for single-keyword Boolean searches.

Xia *et al.* [11] proposed a scheme using the k-nearest neighbor (KNN) technique; this scheme cannot only satisfy multiple keywords search but also sort the search results by the elegance scores between file index and trapdoor. Specifically, the vector space model and the widely used TF*IDF weightage model are combined in index construction and query generation.

They constructed a tree-based index structure and proposed a "Greedy Depth-first Search" algorithm to provide an efficient multi-keyword ranked search. The secure KNN algorithm encrypts the index and query vectors and ensures accurate relevance score calculation between the encrypted index and query vectors. However, the search time increases linearly with the increase of matrix dimension due to the generation of indices and trapdoors.

III. The proposed model

In this section, it is described the proposed cloud storage-based searchable and sharing scheme for encrypted data. It adds searchable functionality with significant improvement to the system to achieve secure search, file-assured deletion, and secure sharing of encrypted data.

A. *System Model*

The system model of the proposed scheme is shown in Fig. 1. This scheme comprises the following entities, Data Owner (DO), Data User (DU), Cloud storage (CS), and Trusted Third Party (TTP). The system model of the proposed scheme comprises the following entities.

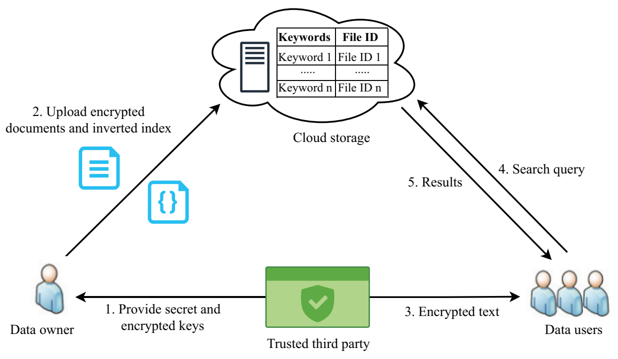

Fig. 1. System model of the proposed scheme.

- **Data Owner (DO):** The DO chooses keywords for documents to search on a specific folder. A secret key is then used to create the corresponding index according to the keywords. Afterward, the DO uploads the encrypted data in that folder with the corresponding index to the CS. The encrypted data consists of two parts: the index of encrypted keywords and the encrypted documents.
- **Data User (DU):** The DU sends a plaintext keyword to the TTP to create an encrypted keyword. In the meantime, the DU downloads the encrypted inverted index from the CS and searches locally with the encrypted keyword. If the system finds a relevant document, it returns the FileID. The DU downloads the encrypted document from the CS by providing FileID. After that, the DU sends the FileID to the TTP along with RSA generated public key. The TTP sends back the decryption key by encrypting it with the public key of the DU. The DU first decrypts the document's decryption key using the private key then decrypts the document.
- **Cloud Storage (CS):** The CS offers storage services for the data owners and users. It is an intermediate entity that stores encrypted documents and inverted index from the DO and searches over the encrypted document by FileID. A matched document will be returned when the DO or DU submits a request to the CS using a specific FileID. The FileID is unique for each document. So, it will return a specific document based on the request.
- **Trusted Third Party (TTP):** The TTP generates the secret and encryption keys to provide the DO for building an inverted index. The TTP is responsible for constructing encrypted text when the DO or DU requests for searching. Also, after submitting the FileID by the DO or DU, TTP verifies the stored FileID of the document. The TTP sends the decryption key of the encrypted document over a secure channel to the DU. One more responsibility of TTP is to control the local server. The TTP's responsibility to handle the local server makes the system less complex.

B. Notations

The notations used in this paper are given in Table I for clarity and readability.

Details of the terms are given below.

- **Setup** (λ): It is run by the TTP. It takes a security parameter λ and outputs the secret key SK, randomly generated 256-bit AES $R_1, \cdots, R_n$ keys for encrypting documents.
- **Encrypt data** (F, R_n): The DO encrypts each document F with the random key R_n and uploads the encrypted documents C to the cloud. The CS will return the FID of the documents.
- **Generate index** (SK, I): The DO chooses random keywords W for each plaintext document F and generates an inverted index I. It requires the secret key SK and FID as inputs to generate the encrypted inverted index I' and upload it to the CS.

TABLE I
NOTATIONS USED FOR THE PROPOSED MODEL

Notations	Descriptions
F	The plaintext collection of n documents
n	The total number of documents in F
F_i	A sequence of documents where F_i is the ID of the i^{th} document $F = \{F_1, F_2, \cdots, F_n\}$
C	The encrypted document collection stored in the cloud storage, C=$\{C_1, C_2, \cdots, C_n\}$
FID	The cloud storage generates the file ID of the document.
W	The set of keywords from F, as a dictionary W =$\{w_1, w_2, \cdots, w_m\}$
m	The total number of keywords in W
T	A plain text keyword by which the DO or DU can search
W_c	The encrypted keyword text to search on the index
I	The form of inverted index I=$\{I_1, I_2, \cdots, I_m\}$ for document collection F.
I′	The searchable encrypted inverted index generated from I
F_w	The set of all files containing the keyword W_c

- **Store** (R_n, FID): The DO stores the random AES R_n keys for each document and FID to the TTP.
- **Generate encrypted keyword** (SK, w): The DU requests TTP to make an encrypted keyword text T of the word w for retrieving the documents from the CS. It outputs an encrypted keyword W_c query by which the DU can search for an encrypted document.
- **Search** (I', W_c): It takes the encrypted keyword W_c as input query and encrypted index I', and returns FID of F_w if the query W_c matches with any keywords from the encrypted inverted index I'.
- **Decrypt data** (C, R_n): The DO or DU decrypts the encrypted document C with the decryption key R_n.

C. Inverted Index

Here it is discussed how the inverted index method is used in the proposed model. For example, if it is considered an inverted index where the plaintext keywords "Headache", "Cold", and "Allergies" are selected from the document ID $D_1, \cdots, D_{10}$ randomly, which is listed in Table II.

TABLE II
INVERTED INDEX OF PLAIN TEXT DATA

Plain text Keywords	Document ID
Headache	D_3, D_5, D_7, D_{10}
Cold	D_5, D_7
...	...
Allergies	D_2, D_3, D_6

A secret key SK is used to encrypt only the keywords, and $R_1, R_2, \cdots, R_n$ 256 bits symmetric encryption keys are used to encrypt the related document in the inverted index listed in Table III.

The DU must download the encrypted inverted index locally to look for the encrypted document. The DU can search on the encrypted index by using the encrypted keyword "Cold" which is encrypted with the secret key SK. If there is any match with the encrypted keyword, the index will return all the documents based on the keyword, i.e., D_5, D_7, to the DU. As the document is encrypted with corresponding random keys

TABLE III
INVERTED INDEX OF ENCRYPTED DATA

Encrypted Keyword	Document ID
874f718fb3df	D_3, D_5, D_7, D_{10}
26b94b7c17a	D_5, D_7
...	...
9619da7d95c	D_2, D_3, D_6

R_5 and R_7, it will return the File ID, which was stored in the cloud server, shown in Table IV.

TABLE IV
ENCRYPTED RESULT BASED ON SEARCH

Keyword Matching	Document ID	File ID
Cold : 26b94b7c17a	D_5	1IVBMUp9xNYJTa4MFKFiY
Cold : 26b94b7c17a	D_7	1ezIJxI5NmkmzjuvtRmwgPG

D. Document Access Scenarios

Data owners and Data users can access previously stored cloud data, and based on that, two scenarios are considered as given below.

- **Scenario 1: Data owner searchable encryption:** The DO outsources the encrypted documents to the CS, search with keywords on those encrypted documents, and gets the result as shown in Fig. 2.
- **Scenario 2: Data user searchable encryption:** The DO outsources the encrypted documents in a folder and shares the folder with the DU by email. The DU searches by keyword on those encrypted documents and decrypts documents with help from the TTP, as shown in Fig. 3.

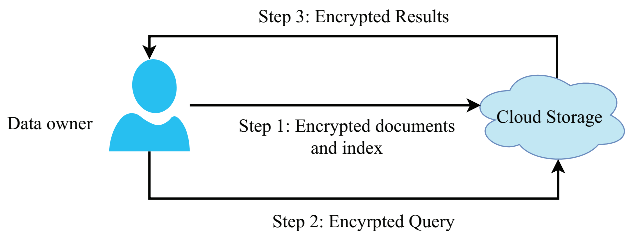
Step 3: Encrypted Results
Data owner
Step 1: Encrypted documents and index
Cloud Storage
Step 2: Encyrpted Query

Fig. 2. Data owner searchable encryption

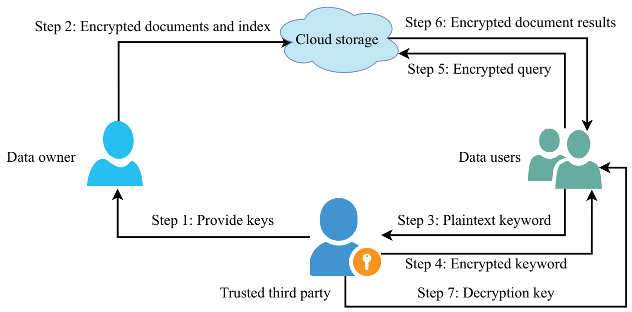
Step 2: Encrypted documents and index
Cloud storage
Step 6: Encrypted document results
Step 5: Encrypted query
Data owner
Data users
Step 1: Provide keys
Step 3: Plaintext keyword
Step 4: Encrypted keyword
Trusted third party
Step 7: Decryption key

Fig. 3. Data user searchable encryption

In the above two scenarios, the steps such as key setup, encryption, index generation, search, download, and decryption are used. The details of these steps are explained below based on DO and DU as follows.

E. Data owner searchable encryption

Here, it is described scenario 1, where DO follows some steps to store and search for encrypted data, which is given below.

1) Setup and Key Generations: The setup and key generation process are discussed below, shown in Fig. 4.

(a) The DO creates a folder in CS to store multiple documents. The CS returns a folder ID to the DO.
(b) The DO requests the TTP by folder ID to get the secret key SK for generating keyword index I and AES keys $R_1, \cdots, R_n$ for encrypting the documents $F_1, \cdots, F_n$.
(c) The TTP takes a security parameter as input and generates the secret key SK and 256-bit AES keys $R_1, \cdots, R_n$ based on the folder ID.
(d) The TTP sends those keys to the DO to encrypt the keywords and documents.

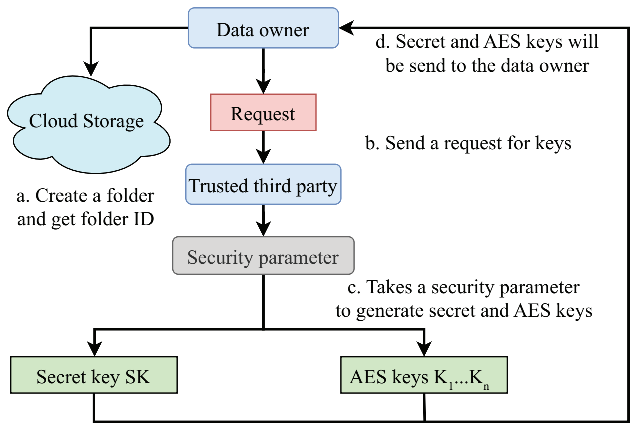
Data owner
d. Secret and AES keys will be send to the data owner
Cloud Storage
Request
b. Send a request for keys
a. Create a folder and get folder ID
Trusted third party
Security parameter
c. Takes a security parameter to generate secret and AES keys
Secret key SK
AES keys K_1...K_n

Fig. 4. Setup and key generate process

2) Encrypt and Upload: The encrypt and upload process of the scheme are discussed below, shown in Fig. 5.

(a) The DO encrypts the documents using AES $R_1, \cdots, R_n$ keys and uploads those encrypted documents to the CS.
(b) The CS returns the file ID of each document to the DO.
(c) After getting the file ID, the DO chooses keywords independently for each file ID and generates an inverted index.
(d) The DO encrypts those keywords using the secret key SK and uploads the inverted index to the CS.

3) Store: The store process of the scheme is discussed below, shown in Fig. 6.

(a) The DO stores the folder ID and AES key of each file ID to the TTP after uploading the encrypted documents and inverted index to the CS.
(b) The TTP saves all this information in a database. It is needed when the DO searches for a document or shares the folder with data users.

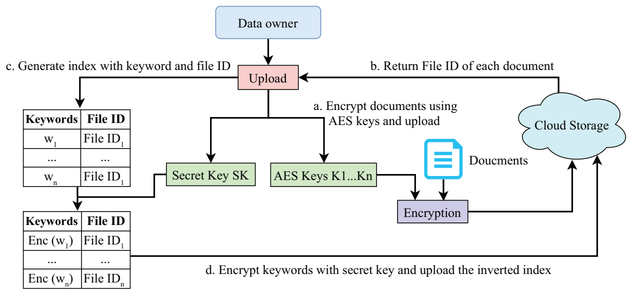
Data owner
c. Generate index with keyword and file ID
Upload
b. Return File ID of each document
a. Encrypt documents using AES keys and upload
Cloud Storage
Keywords | File ID
w_1 | File ID_1
... | ...
w_n | File ID_1
Secret Key SK
AES Keys K1...Kn
Documents
Encryption
Keywords | File ID
Enc (w_1) | File ID_1
... | ...
Enc (w_n) | File ID_n
d. Encrypt keywords with secret key and upload the inverted index

Fig. 5. Encrypt and upload process

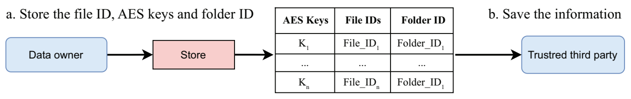
a. Store the file ID, AES keys and folder ID
Data owner
Store
AES Keys | File IDs | Folder ID
K_1 | File_ID_1 | Folder_ID_1
... | ... | ...
K_n | File_ID_n | Folder_ID_1
b. Save the information
Trusted third party

Fig. 6. Store process

4) Search: When the DO wants to search for a document in the folder, it needs to generate an encrypted keyword. Assume that many folders are created in the CS by the DO and uploaded multiple documents. So the DO does not need to store and recognize the secret key of each folder to generate encrypted text to search on the CS and get the file. The search and download process is discussed below, shown in Fig. 7.

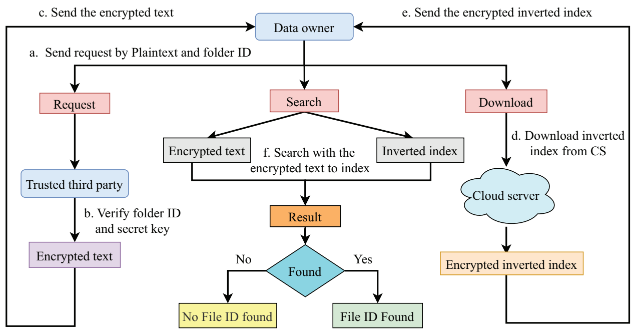
c. Send the encrypted text
Data owner
e. Send the encrypted inverted index
a. Send request by Plaintext and folder ID
Request
Search
Download
Encrypted text
f. Search with the encrypted text to index
Inverted index
d. Download inverted index from CS
Trusted third party
Cloud server
b. Verify folder ID and secret key
Result
Encrypted text
No
Found
Yes
Encrypted inverted index
No File ID found
File ID Found

Fig. 7. Search process

(a) The DO sends a plaintext keyword with the folder ID to the TTP for encryption text by which the DO wants to search.
(b) The TTP checks the folder ID and generates an encrypted text keyword by the secret key of that particular folder for that plaintext.
(c) The TTP sends the encrypted text to the DO.
(d) In the meantime, the DO sends a request for the encrypted inverted index to the CS. The CS does not know which keyword was searched for the specific document in this way.
(e) The CS returns the encrypted inverted index to the DO.
(f) The DO searches with the encrypted text on inverted text. If the encrypted text matches any keyword from the inverted index, it returns the file ID of the document. The scheme can return multiple documents simultaneously if the multiple documents contain the same keyword.

5) Download and decrypt: The data owner must download the encrypted document from cloud storage and follow some steps to decrypt it, which is discussed below, shown in Fig. 8.

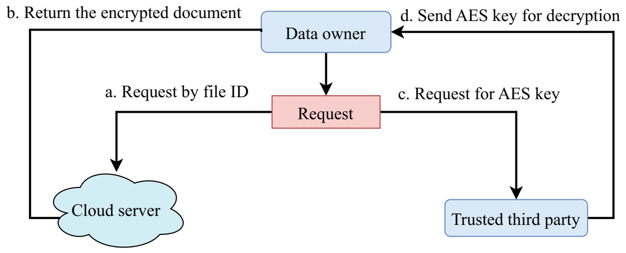

Fig. 8. Download and decrypt process

(a) The DO requests the CS for an encrypted document by file ID.
(b) The CS returns the encrypted document to the DO.
(c) The DO sends the request by file ID to the TTP for the AES key to decrypt the document.
(d) The TTP checks the file ID and sends the AES key to the DO for decryption through a secure channel.
(e) The DO decrypts the document with that particular AES decryption key.

F. Data user searchable encryption

Here, it is described scenario 2, where DO shares the folder with DU and DU follows some steps to search on encrypted data, which are given below.

1) Access: The sharing process of the scheme for DU is discussed below, shown in Fig. 9.

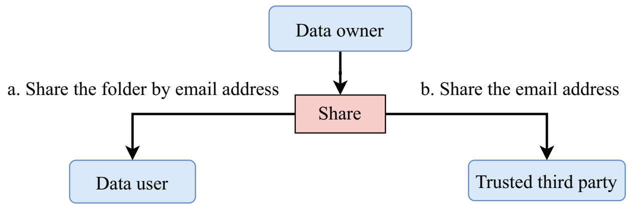
Data owner
a. Share the folder by email address
Share
b. Share the email address
Data user
Trusted third party

Fig. 9. Share process

(a) The DO shares the folder by email address of the DU. The DU gets an email that the DO shared a folder.
(b) The DO also shares the email address of the DU with the TTP with Identity-based encryption. The DU is identified and authorized for data access based on their recognizable identities by the TTP.

2) Search: The keyword search process of the scheme is discussed below, shown in Fig. 10.

(a) The DU does not know the exact keyword by which the DO generated the inverted index. The DU only got folder access from the DO. So, the DU sends a request by choosing any relevant plain text with the document by folder ID to the TTP to generate searchable encryption text.
(b) The TTP verifies the email address of the DU and whether the DU is identified and authorized for data access.

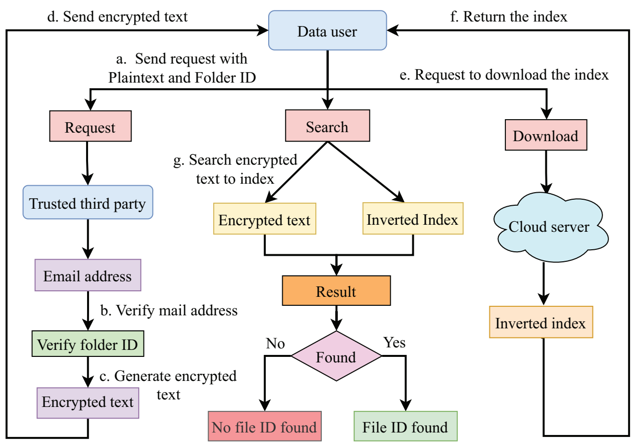

Fig. 10. Search process

(c) The TTP checks the folder ID and generates an encrypted text by the secret key of that particular folder.
(d) The TTP sends the encrypted text to the DU for keyword search.
(e) The DU requests to download the encrypted inverted index of the folder from the CS.
(f) The CS returns the encrypted inverted index to the DU.
(g) The DU searches by encrypted text to the encrypted inverted index. If the keyword is matched from the inverted index, it returns a file ID of the encrypted document.

3) Decryption: The decryption process of the scheme is discussed below, shown in Fig. 11.

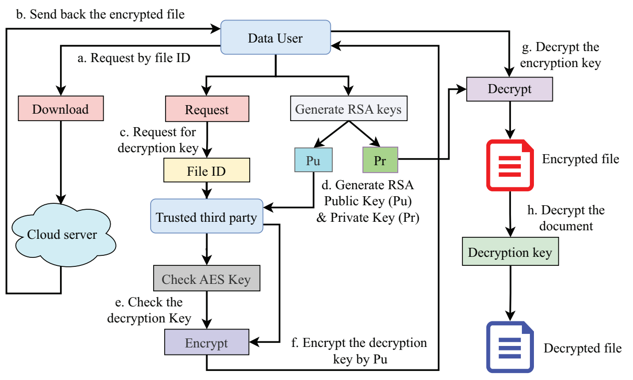
b. Send back the encrypted file
Data User
a. Request by file ID
g. Decrypt the encryption key
Download
Request
Generate RSA keys
Decrypt
c. Request for decryption key
Pu
Pr
Encrypted file
File ID
d. Generate RSA Public Key (Pu) & Private Key (Pr)
h. Decrypt the document
Trusted third party
Cloud server
Decryption key
Check AES Key
e. Check the decryption Key
Encrypt
f. Encrypt the decryption key by Pu
Decrypted file

Fig. 11. Decryption process

(a) The DU sends a request to the CS by file ID for downloading.
(b) The CS sends the encrypted document to the DU.
(c) The DU sends a request to the TTP by that encrypted document's file ID for the decryption key.
(d) The DU generates a pair of random RSA public (Pu) and private key (Pr) and sends the public key (Pu) to the TTP.
(e) The TTP checks the decryption key of the document, which the DO has already stored.
(f) The TTP encrypts the decryption key of the document by the public key (Pu) and sends it to the DU.
(g) The DU decrypts the decryption key of the document with the private key (Pr).
(h) At the end, the DU decrypts the document with the decryption key.

As a part of this work, a web application, "Crypto Search" was implemented as an overlay system on top of a well-known cloud storage domain to demonstrate the proposed model. It exhibits better efficiency than the existing schemes in searching and updating keywords, while it can be practically used in commercial applications. Details of the implementation are not given in this paper due to space limitations.

IV. Conclusions

It has proposed a keyword-based searching and sharing scheme on encrypted cloud storage data. It solves the security problems of data storage, sharing, and keyword-based searching on encrypted data over the cloud. The model is designed in such a way that it is practical, efficient, and user-friendly. However, there are still some limitations to our proposed models, such as if the trusted party is honest and curious about the document, the entity can act as an adversary or intruder to a data user. In that case, the whole folder document becomes vulnerable. The method of keyword searching is limited to the exact keyword search. Any keyword search is a good future extension to this work. The proposed scheme improves the trade-off between search capability and security when storing encrypted data.

References

[1] Yao, L., Weng, J., Yang, A., Liang, X., Wu, Z., Jiang, Z. and Hou, L., 2023. Scalable CCA-secure public-key authenticated encryption with keyword search from ideal lattices in cloud computing. Information Sciences, 624, pp.777-795.
[2] Wu, Q., Lai, T., Zhang, L., Mu, Y. and Rezaeibagha, F., 2022. Blockchain-enabled multi-authorization and multi-cloud attribute-based keyword search over encrypted data in the cloud. Journal of Systems Architecture, 129, p.102569.
[3] Singh, N., Kumar, J., Singh, A.K. and Mohan, A., 2022. Privacy-preserving multi-keyword hybrid search over encrypted data in cloud. Journal of Ambient Intelligence and Humanized Computing, pp.1-14.
[4] Song, D.X., Wagner, D., and Perrig, A., 2000, May. Practical techniques for searches on encrypted data. In Proceeding 2000 IEEE symposium on security and privacy. S and P 2000 (pp. 44-55). IEEE.
[5] Patwary, M.S.A. and Palit, R., 2019, December. A Prototype of a Secured File Storing and Sharing System for Cloud Storage Infrastructure. In 2019 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE) (pp. 1-8). IEEE.
[6] Chaudhari, P. and Das, M.L., 2019. Privacy-preserving searchable encryption with fine-grained access control. IEEE Transactions on Cloud Computing, 9(2), pp.753-762.
[7] Curtmola, R., Garay, J., Kamara, S. and Ostrovsky, R., 2011. Searchable symmetric encryption: improved definitions and efficient constructions. Journal of Computer Security, 19(5), pp.895-934.
[8] Wang, B., Song, W., Lou, W. and Hou, Y.T., 2015, April. Inverted index-based multi-keyword public-key searchable encryption with a strong privacy guarantee. In 2015 IEEE Conference on Computer Communications (INFOCOM) (pp. 2092-2100). IEEE.
[9] Goh, E.J., 2003. Secure indexes. Cryptology ePrint Archive.
[10] Kamara, S. and Papamanthou, C., 2013, April. Parallel and dynamic searchable symmetric encryption. In International conference on financial cryptography and data security (pp. 258-274). Springer, Berlin, Heidelberg.
[11] Xia, Z., Wang, X., Sun, X. and Wang, Q., 2015. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Transactions on Parallel and Distributed systems, 27(2), pp.340-352.