VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Project Report on

Searching and Indexing over Encrypted Data on Cloud

Submitted by

Aakash Goplani (D17A-29)
Sneha Kukreja (D17A-52)
Jyoti Vaswani (D17A-80)

Project Mentor

Prof Anjali Yeole.

In partial fulfillment of the Fourth Year, Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai

Academic Year 2014-2015

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Project Report on

Searching and Indexing over Encrypted Data on Cloud

Certificate Of Approval

This is to certify that ***Aakash Goplani, Sneha Kukreja and Jyoti Vaswani*** of Fourth Year Computer Engineering studying under the University of Mumbai have satisfactorily completed the project on "***Searching and Indexing over Encrypted Data on Cloud***" as a part of their coursework of PROJECT-II for Semester-VIII under the guidance of their mentor ***Prof. Anjali Yeole*** in the year 2014-2015.

Date

Head of the Department
(Dr. (Mrs.) Nupur Giri)

PEO	Outcomes	Grade
I,II,III,IV	a,b,c,d,e,f,g,h,i,j	

Project Mentor
(Prof. Anjali Yeole)

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Project Report on

Searching and Indexing over Encrypted Data on Cloud

Submitted by

Aakash Goplani (D17A-29)

Sneha Kukreja (D17A-52)

Jyoti Vaswani (D17A-80)

Project Mentor

Prof Anjali Yeole.

In partial fulfillment of the Fourth Year, Bachelor of Engineering (B.E.) Degree in Computer Engineering at the University of Mumbai

Academic Year 2014-2015

____

Date

____________________	____________________
Internal Examiner	External Examiner

____________________	____________________	____________________
Project Mentor (Mrs. Anjali Yeole)	Head of the Department (Dr. (Mrs.) Nupur Giri)	Principal (Dr. (Mrs.) J. M. Nair)

ACKNOWLEDGEMENT

First of all, we thank our department (The Computer Department of V.E.S.I.T.) for providing us with the opportunity to work on this project.

We would like to take this opportunity to thank all our teachers who have provided their valuable support and guidance to us during the course of this project.

A special thanks to our mentor **Prof. Anjali Yeole** who provided utmost help and support in terms of material and guidance. She has been a great support since the early stages of the project in developing the idea as well as finding out the resources. We would like to thank her for leading us into the right direction throughout the course of this project.

ABSTRACT

We present a scheme that discusses keyword search over an encrypted cloud data. The data that has to be outsourced is encrypted using symmetric encryption algorithm for data confidentiality.

The index file of the keyword set that has to be searched is outsourced to the local trusted server where the keyword set that is generated from the data files is also stored. This is done so that any un-trusted server cannot learn about the data with the help of the index formed.

The index is created with the help of algorithms which matches the pre-defined set of keywords with information in the data files to index them and store relevant data.

Whenever the user searches for a keyword, the request is sent to the local trusted server and the indexed data is referred. The files are listed based on the certain relevance criteria.

User requests for the required files to the un-trusted server. The parameters required for ranking is got from the data stored while indexing. Based on the ranking, the files are retrieved from the un-trusted server and displayed to the user.

The proposed system can be extended to support Boolean search and Fuzzy keyword search techniques.

TABLE OF CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

1.1 MOTIVATION

As data sizes grow, so does the need for efficient search. Storing a back-catalogue of 5,000 emails serves no purpose, aside from filling up your hard disk, unless they can be easilsy searched. This becomes exponentially more apparent when dealing with company-wide email systems and other large scale data collections. These systems are also inherently distributed in nature, either internally to the company (perhaps in the main office buildings), or more likely, outsourced (maybe to a local datacenter, or even abroad). If we want data to remain secure, especially if outsourced to a third party, we need to use encryption. Once the data is encrypted and outsourced to a far-off datacenter we need to be able to access it. If we need to perform a search over the encrypted data, we have two options:

1) We can download the entire data set, decrypt it and search it client-side.

2) We can give the remote server the secret key we used to encrypt the data and let the server decrypt and search it.

In this scenario, both of the 'solutions' described above have fairly obvious ramifications.

1) Downloading all data each time you need particular data leads to an impossibly large amount of communication overhead.

2) If secret key is provided to server to decrypt document and search over it, it is required to search data center. Knowing the key, a rogue datacenter admin could perform a number of harmful acts.

1.2 PROBLEM DEFINITION

As Cloud Computing is one of the biggest and trending technologies, more and more Companies are outsourcing their databases on Cloud in order to lower the cost of maintaining hardware. But the biggest concern about Cloud is Security and Privacy. In order to solve this problem sensitive data is encrypted before outsourcing. Searching particular data over encrypted data is a challenging task. In this project we propose to study and analyze different searching and indexing techniques on encrypted data. Also based on efficiency in space and time we propose to develop an efficient application which will search for keywords in encrypted documents and documents which contain that keyword will be returned. Purpose of such application is to allow authenticated user to search for documents without providing any knowledge to third party which stores the data. It allows user to search over encrypted data without the need for decrypting it.

1.3 RELEVANCE OF THE PROJECT

Cloud Computing is an emerging technology, hence it has become the common practice to use cloud, since it not only provides cost efficiency but also on demand high quality service. The data on Cloud could be anything ranging from Emails, Personal Files, University records to Government data, etc. Also the users of this technology can range from Private sectors i.e. banks, universities, companies to Public Sector.

These users outsource their databases on Cloud to enjoy all the services. But as Cloud is hosted by a third party, data stored on cloud cannot be completely trusted for privacy and security. It follows that sensitive data usually should be encrypted prior to outsourcing for data privacy and preventing unsolicited accesses. Database encryption introduces an additional layer to conventional network and application security solutions and prevents exposure of sensitive information even if the raw data is compromised. Database encryption prevents unauthorized users from viewing sensitive data in the database and, it allows database administrators to perform their tasks without having access to sensitive information. Furthermore, it protects data integrity, as unauthorized modifications can easily be detected .

However as a result of encryption, effective data utilization becomes a challenging task. Moreover, in Cloud Computing, data owners may share their outsourced data with a large number of users. The individual users might want to only retrieve certain specific data files they are interested in during a given session, but encryption of data makes searching of documents required difficult.

Hence efficient search technique is needed to be implemented over encrypted data.

1.4 METHODOLOGY USED

1.4.1 Encryption Algorithm AES

AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification *per se* is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

1.4.2 PHP

The PHP Hypertext Preprocessor (PHP) allows web developers to create dynamic content that interacts with databases. Common uses and characteristics of php are: PHP performs system functions, handles form. It can add, delete and modify elements in database, access cookies variables etc.

1.4.3 Keyword Generation

A simple key word generation algorithm in php that reads the entire file and delete all the stop words (like I, am, those, these… etc) and the remaining words are considered as keywords. The occurrence of each keyword is calculated and displayed as frequency of that particular character.

1.4.4 String Matching Algorithm

The search algorithm implemented in php involves users query to be splitted up into keywords after removing all the stop words from the query. For every indexed word in the database that matches our search term, we will give it a score. Based on ranking function, the top-k files will be retrieved and user can download them accordingly.

1.4.5 Ranking

TF-IDF is used as Ranking Function.

Tf-idf stands for *term frequency-inverse document frequency*, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

Typically, the tf-idf weight is composed by two terms: the first computes the normalized Term Frequency (TF), i.e. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

- **TF: Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (the total number of terms in the document) as a way of normalization:
  TF (t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- **IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important.
  IDF (t) = log_e (Total number of documents / Number of documents with term t in it).

Example: Consider a document containing 100 words wherein the word *cat* appears 3 times. The term frequency (i.e., tf) for *cat* is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word *cat* appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log (10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12.

2. LITERATURE SURVEY

2.1 RELATED WORKS AND BACKGROUND

Searchable encryption is not a new concept but all current methods have failed in various aspects that keep them from becoming common or mainstream. Even ranked word proximity searches, a search that ranks results based on how close the query keywords are together, has not been fully implemented by previous research.

Song et al. proposed a searchable encryption scheme based on a symmetric key. The scheme involved provably secure, query isolation for searches, controlled searching, and support hidden query. *Drawbacks*: Case insensitivity, regular expression and sub matches are not supported. Speed of searching and total space required is huge [1].

Eu-Jin Goh proposed an index searching algorithm based on a bloom filter. Their scheme reduced the computational overhead of loading an index and searching for files. *Drawbacks*: Bloom filters result in false positives; updating procedure lacks security analysis, Security model not satisfactory for Boolean searches, unclear experimental evaluation [2].

PEKS: Public Encryption Keyword Search makes use of public key cipher technique. This technique focuses on refreshing keywords, removing secure channel and processing multiple keywords. *Drawbacks:* List of keyword has to be determined carefully in order to keep length of message down. Public key algorithms require large prime numbers to be calculated in order to generate usable keys, so this process is potentially very time consuming [3].

PKIS: Practical Keyword Index Search on cloud datacenter focuses on group search over encrypted database. It involves two schemes: PKIS-1 and PKIS-2. PKIS provide practical, realistic, and secure solutions over the encrypted DB. *Drawbacks:* The common keywords in different documents for certain group have the same index values, which leads to Brute Force attacks [4].

APKS: Authorized private keyword search over encrypted data in cloud computing deals with multi-keyword search. Multi-dimensional query are converted to its CNF (Conjunctive Normal Form) formula and are arranged in a hierarchical way. *Drawbacks:* APKS does not prevent keyword attack [5].

Multi-keyword Ranked Search over Encrypted Cloud Data (MRSE) uses "co-ordinate matching" principle i.e. as many matches as possible, it is an efficient principle among multi-keyword semantics to refine the result relevance. *Drawbacks*: Even though keywords are protected by trapdoors server can do some statistical analysis over search result. Server can generate trapdoor for subset of any multi keyword trapdoor request [6].

2.2 ENCRYPTED PHRASE SEARCHING IN THE CLOUD

This technique allows both encrypted phrase searches and proximity ranked multi-keyword searches to encrypted datasets on untrusted cloud [7].

1. The client sends a plaintext search query to a trusted client-side server.
2. The Client-side server encrypts all keywords in the search query individually using symmetric-key encryption; it then truncates the encrypted keywords to a set number of bits to improve security by allowing for collisions, and queries the untrusted cloud server for the documents containing the set of truncated encrypted keywords.
3. The cloud server does a database query of its encrypted index and returns to the client-side server encrypted data that corresponds to document paths, truncated encrypted keyword index offset, and encrypted keyword locations.
4. The client-side server decrypts this data first. From the newly decrypted keyword index offset it can then determine which returned results are actually for the keywords searched and which are simply collisions. It discards those collisions and filters and/or ranks the pertinent returned documents based on relevant keyword locations and frequency. Finally, it sends this ranked listing to the original client.

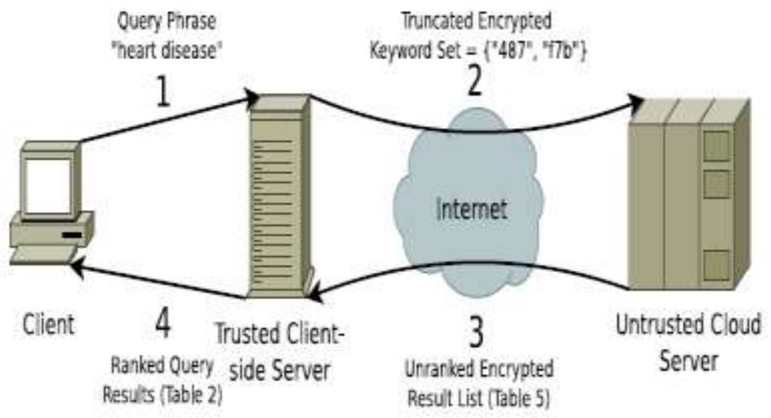

Query Phrase "heart disease"
1
Truncated Encrypted Keyword Set = {"487", "f7b"}
2
Internet
Client
4
Ranked Query Results (Table 2)
Trusted Client-side Server
3
Unranked Encrypted Result List (Table 5)
Untrusted Cloud Server

Fig. 1: Encrypted Phrase Search.

Disadvantages:

1. Does not support fuzzy keyword search.
2. Does not support synonym query.
3. Not suitable for multi-user environment.

2.3 PROGRAMMABLE ORDER-PRESERVING SECURE INDEX FOR ENCRYPTED DATABASE QUERY

This technique proposes an order-preserving scheme for indexing encrypted data, which facilitates the range queries over encrypted databases. The scheme is secure since it randomizes each index with noises, such that the original data cannot be recovered from indexes. Moreover, scheme allows the programmability of basic indexing expressions and thus the distribution of the original data can be hidden from the indexes.

In this scheme, the *i*th value in the plaintext domain is mapped to the *i*th value in the cipher text domain, such that the order between plaintexts is preserved between cipher texts.

The scheme is built over the simple linear expressions of the form *a* * + *b*. The form of the expressions is public, however the coefficients *a* and *b* are kept secret (not known by attackers). Based on the linear expressions, the indexing scheme maps an input value to *a* * + *b* + noise, where noise is a random value. The noise is carefully selected, such that the order of input values is preserved. Indexing scheme allows the programmability of basic indexing expressions (i.e., the linear expressions). Users can make an indexing program that deals with different input values with different indexing expressions. On the one hand, the programmability improves the robustness of the scheme against brute-force attacks since there are more indexing expressions to attack. On the other hand, the programmability can help decouple the distributions of input values and indexes. When a single linear expression is used to index all input values, the distribution of indexes is identical to the distribution of input values. This problem can be addressed by designing appropriate indexing programs [8].

2.4 PRIVACY- PRESERVING KEYWORD-BASED SEMANTIC SEARCH OVER ENCRYPTED CLOUD DATA

The Semantic search technique reinforces the system usability by returning the exactly matched files and the files including the terms semantically similar to the query keyword. The co-occurrence of terms is used as the metric to evaluate the semantic distance between terms in semantic relationship library (SRL). The SRL is constructed as a weighted graph structure [9].

1. Data owner has a collection of text files the owner then constructs a metadata for each file and outsources the encrypted metadata set to the private cloud server. The text files are encrypted by using traditional symmetric encryption algorithm and uploaded to the public cloud server.
2. Private cloud server constructs the inverted index and semantic relationship library using metadata set provided by data user. Then the Inverted index is outsourced to the public cloud server for retrieval.
3. The authorized data users provide the search trapdoor to the private cloud server. Here, the authorization between the data owner and users is appropriately done.
4. Upon receiving the request, the private cloud server extends the query keyword upon SRL and uploads the extended query keywords set to the public cloud.
5. Upon receiving the search request, the public cloud retrieves the index, and returns the matching files to the user in order.
6. Finally, the access control mechanism is employed to manage the capability of the user to decrypt the received files.

Disadvantages:
1. Does not support multi-user environment.
2. The size of range cannot be unboundedly large. So the range size | R | should be properly tradeoff between randomness and efficiency.
3. Does not support synonym query.

2.5 VERIFIABLE ATTRIBUTE-BASED KEYWORD SEARCH WITH FINE-GRAINED OWNER-ENFORCED SEARCH AUTHORIZATION IN THE CLOUD

The outsourced dataset can be contributed from multiple owners and are searchable by multiple users, i.e. multi-user multi-contributor case. The attribute-based keyword search scheme with efficient user revocation (ABKS-UR) enables scalable fine-grained (i.e. file-level) search authorization (*fine-grained owner-enforced search authorization*) using attribute based (CP-ABE) technique. In the CP-ABE technique specifically, for each file, the data owner generates an access-policy-protected secure index, where the access structure is expressed as a series of AND gates. Only authorized users with attributes satisfying the access policies can obtain matching result. Users can generate their own search capabilities without relying on an always online trusted authority. The scheme enables authenticity check over the returned search results. The index is encrypted with an access structure rather than public or secret keys based on the attributes (properties of users) of authorized users, which makes the proposed scheme more scalable and suitable for the large scale file sharing system. In this key policy attribute-based encryption (KP-ABE) scheme, cipher text can be decrypted only if the attributes that are used for encryption satisfy the access structure on the user private key [10].

1. The data owner generates the secure indexes with attribute-based access policies before outsourcing them along with the encrypted data into the CS (cloud server).
2. To search the datasets contributed from various data owners, a data user generates a trapdoor of keyword of interest using his private key and submits it to the CS. So as to accelerate the entire search process, enforce the coarse-grained *dataset search authorization* with the *per-dataset* user list such that search does not need to go to a particular dataset if the user is not on the corresponding user list.
3. The fine grained *file-level* search authorization is applied on the authorized dataset in the sense that only users, who are granted to access a particular file, can search this file for the intended keyword. The data owner defines an access policy for each uploaded file.
4. The CS will search the corresponding datasets and return the valid search result to the user if and only if the attributes of the user on the trapdoor satisfy the access policies of the secure indexes of the returned files, and the intended keyword is found in these files.

Disadvantages:
1. Does not support synonym query.
2. Does not support semantics based keyword search.

2.6 PRIVACY-PRESERVING MULTI-KEYWORD FUZZY SEARCH OVER ENCRYPTED DATA IN THE CLOUD

The multi-keyword fuzzy search scheme exploits the locality-sensitive hashing technique. This scheme eliminates the requirement of a predefined keyword dictionary [11].

1. To outsource a set of files to the cloud, the *data owner* builds a secure searchable index for the file set and then uploads the encrypted files, together with the secure index, to the *cloud server*.
2. To search over the encrypted files, an authorized *user* first obtains the trapdoor, i.e., the “encrypted” version of search keyword(s), from the data owner, and then submits the trapdoor to the cloud server.
3. Upon receiving the trapdoor, the cloud server executes the search algorithm over the secure indexes and returns the matched files to the user as the search result.

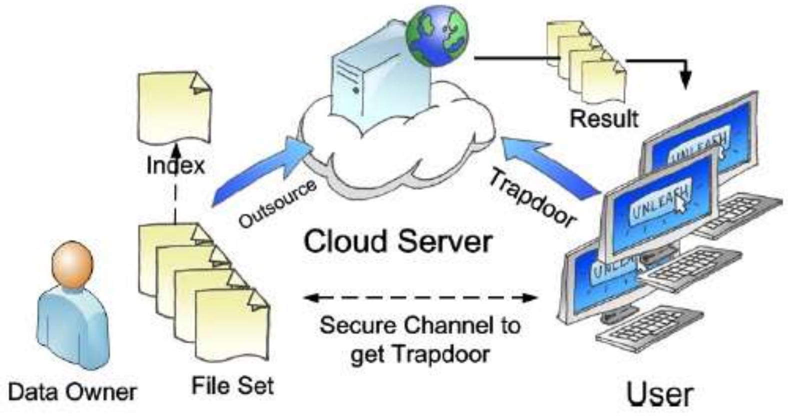


Fig. 2: Architecture representation of multi-keyword fuzzy search.

Disadvantages:
1. The technique is vulnerable to predictive attacks under known cipher text model.
2. Does not support synonym query.
3. Not suitable for multi-user environment.
4. Does not allow semantics based search.

2.7 MULTI-KEYWORD RANKED SEARCH OVER ENCRYPTED CLOUD DATA SUPPORTING SYNONYM QUERY

The technique proposes a semantics-based multi-keyword ranked search scheme over encrypted cloud data which supports synonym query. The search results can be achieved when authorized cloud customers input the synonyms of the predefined keywords, not the exact or fuzzy matching keywords, due to the possible synonym substitution and/or her lack of exact knowledge about the data [12].

Disadvantage:
1. Does not support syntactic transformation, anaphora resolution and other natural language processing technology.

3. REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

1. Less Time required to process user requests.

2. Less Space Required.

3. Low false penalty.

4. Allow sub match, case-insensitive match of keyword.

5. Cost efficient searching.

6. Reveal less information to third party which stores the data.

7. User friendly application GUI.

3.2 NON-FUNCTIONAL REQUIREMENTS

1. Extensibility (adding features, and carry-forward of customizations at next major version upgrade)

2. Usability (Authenticated users can easily avail the features of this application)

3. Testability (An intensive series of tests can be applied to check if all its functional requirements are met or not)

4. Secure and Reliable (Authentication and Validation parameters are added which makes system secure. A backup database is provided so that system continued to function even in case of failure.)

3.3 CONSTRAINTS

1. The architecture should be highly scalable and the application should be tuned for better performance at the peak time.

2. Application should be highly customizable and flexible enough to easily deploy.

3.4 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

• System: Pentium IV 2.4 GHz.

• Hard Disk: 40 GB.

• Floppy Drive: 1.44 Mb.

• Monitor: 14’ Color Monitor.

• Mouse: Optical Mouse.

• Ram: 512 Mb.

• Keyboard: 101 Keyboards.

Software Requirements:

• Operating system: Windows XP, 7, 8.

• Coding Language: PHP, Java Script, CSS, HTML, AJAX

3.5 SYSTEM BLOCK DIAGRAM

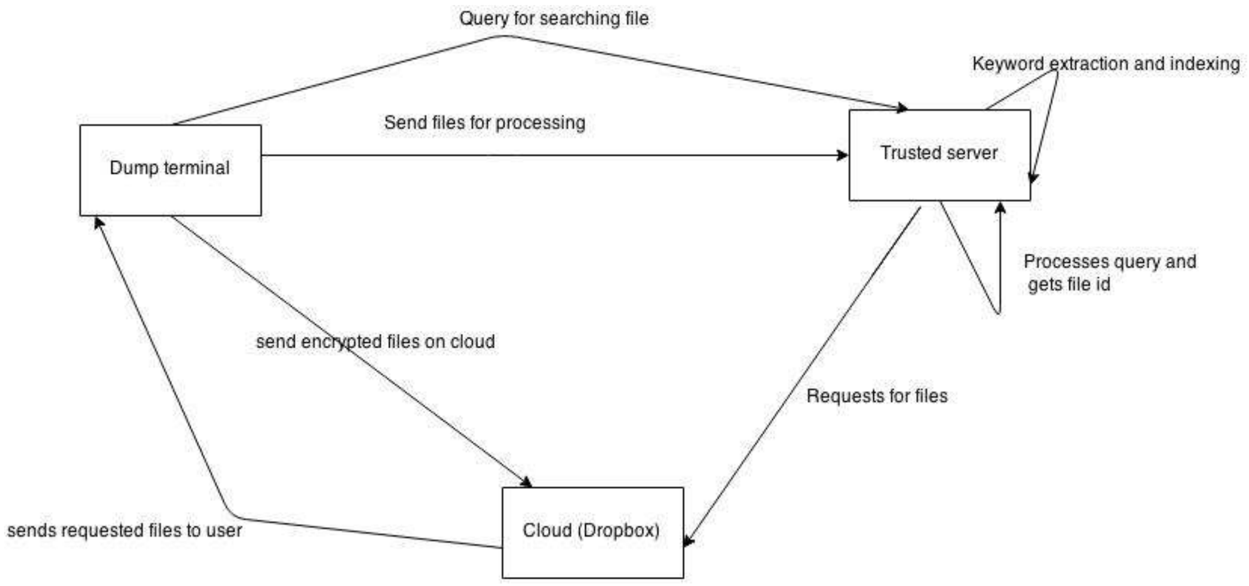

Query for searching file
Keyword extraction and indexing
Send files for processing
Dump terminal
Trusted server
Processes query and gets file id
send encrypted files on cloud
Requests for files
sends requested files to user
Cloud (Dropbox)

Fig. 3: Block diagram of the system

4. PROPOSED DESIGN

4.1 SYSTEM DESIGN

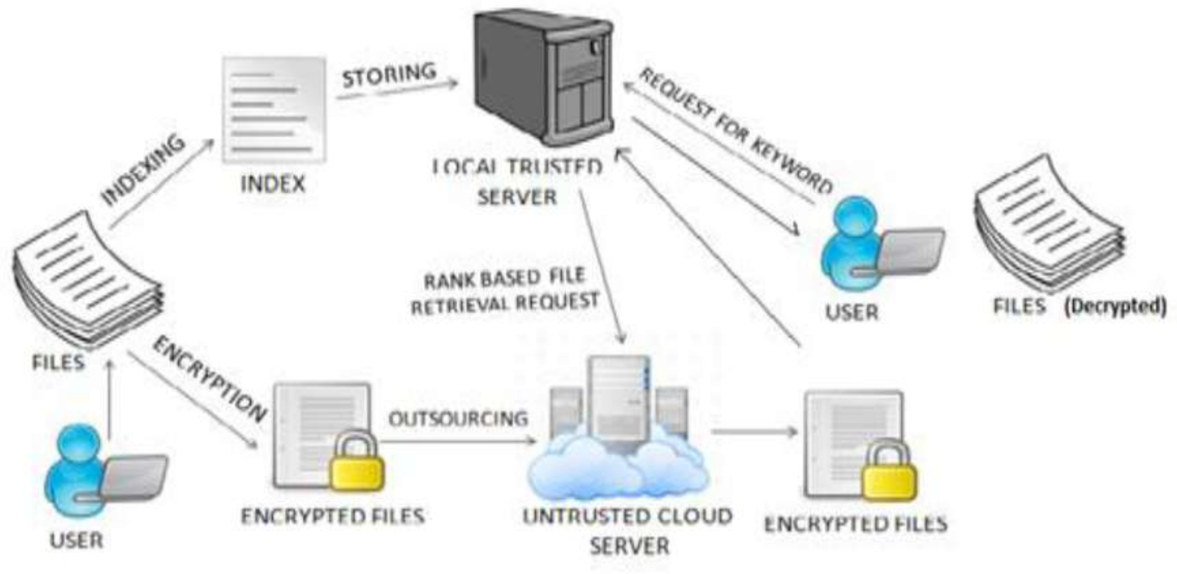


Fig. 4: Diagrammatic Representation of the system

4.2 DETAILED DESIGN

4.2.1 Use Case

The following use case diagram depicts the working of different use cases and scenarios of above mentioned algorithm which is developed for this project. It gives the detailed summary of how the algorithm actually works.

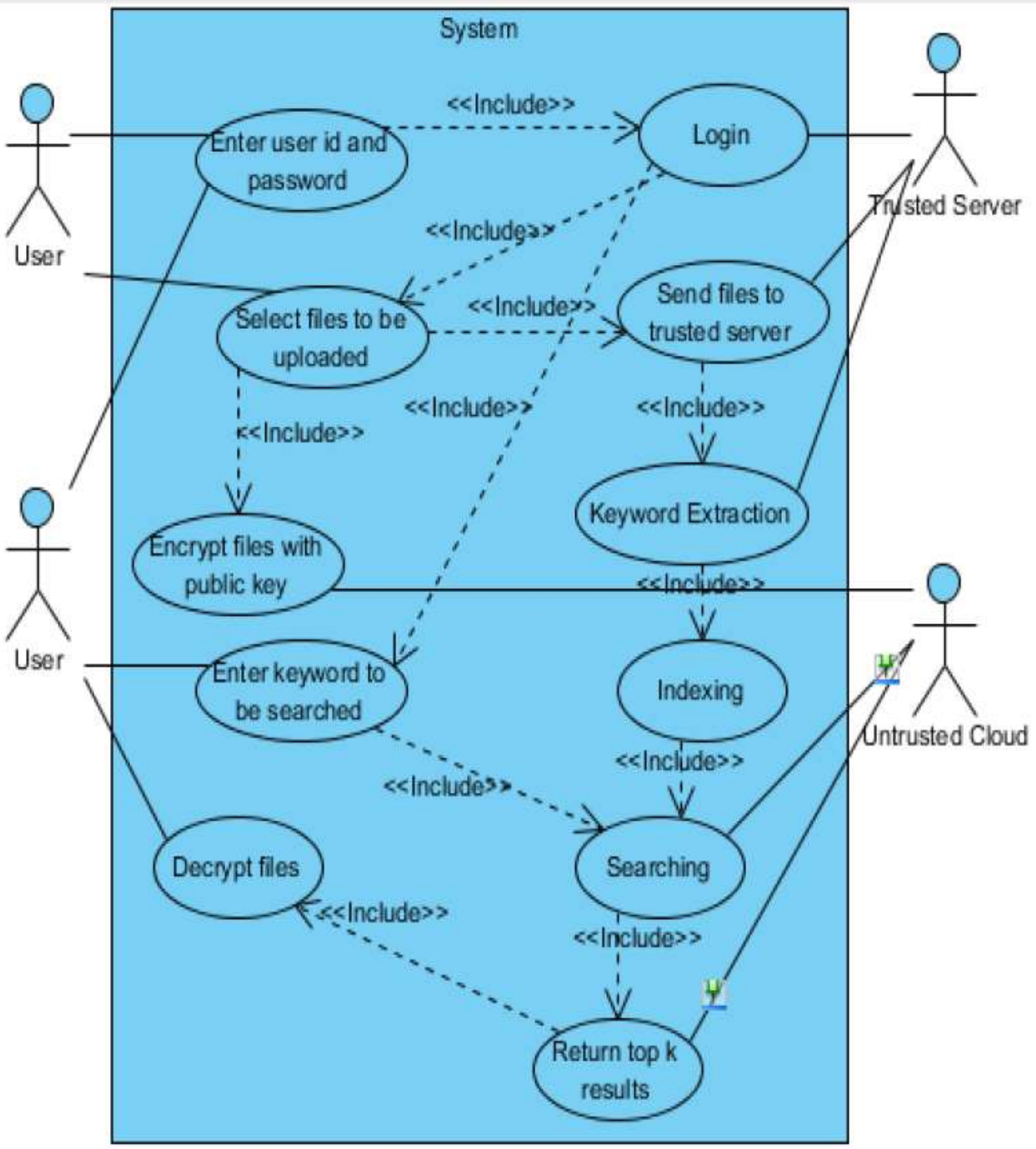

System
User
User
Trusted Server
Untrusted Cloud
Enter user id and password
Login
Select files to be uploaded
Send files to trusted server
Encrypt files with public key
Keyword Extraction
Enter keyword to be searched
Indexing
Decrypt files
Searching
Return top k results
<<Include>>

Fig. 5: Use case diagram of the system

4.2.2 State Diagram (for the system)

The following state diagram depicts the internal working of above mentioned algorithm which is developed for this project. It gives the detailed summary of how the algorithm actually works

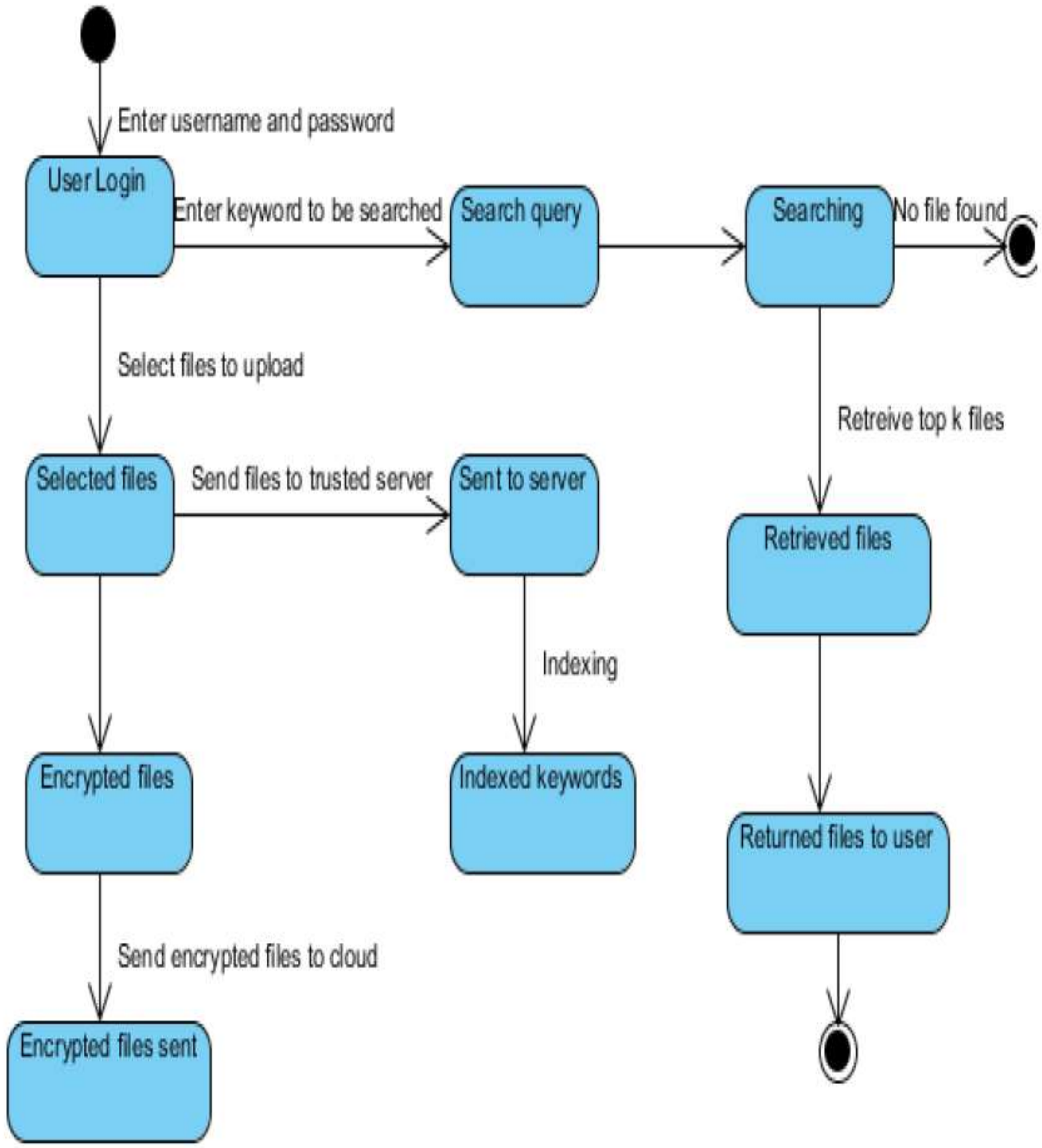


Fig. 6: State transition diagram of the system

4.2.3 Sequence Diagram (for the system)

The following sequence diagram shows the sequence of events that happen while the system is under execution.

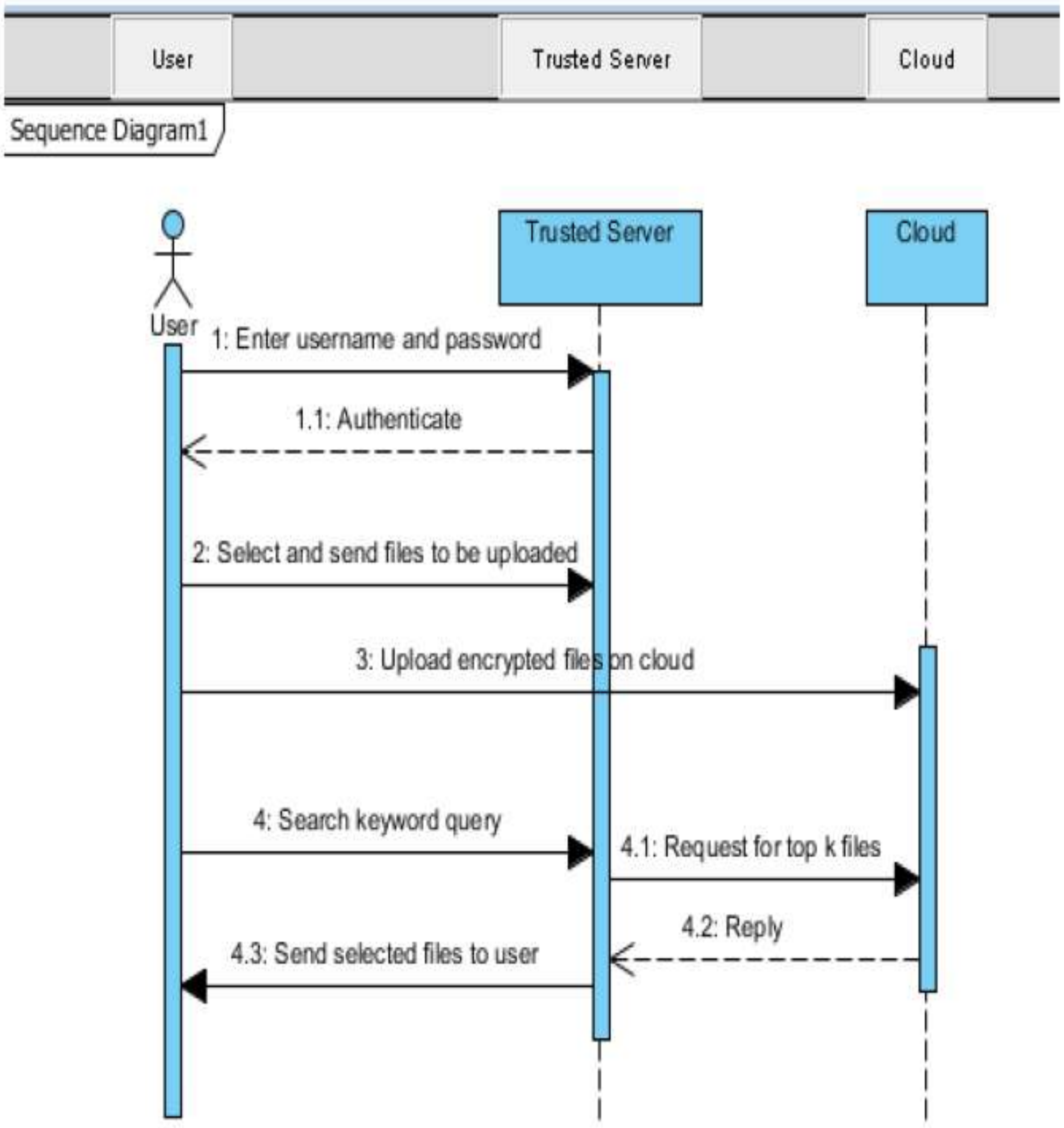

User
Trusted Server
Cloud
Sequence Diagram1
User
Trusted Server
Cloud
1: Enter username and password
1.1: Authenticate
2: Select and send files to be uploaded
3: Upload encrypted files on cloud
4: Search keyword query
4.1: Request for top k files
4.2: Reply
4.3: Send selected files to user

Fig. 7: Sequence diagram of the system

4.2.4 Dataflow Diagram (for the system)

The following diagram represents the dataflow diagram of the entire system and depicts it's working. It shows how data flows between entities and processes.

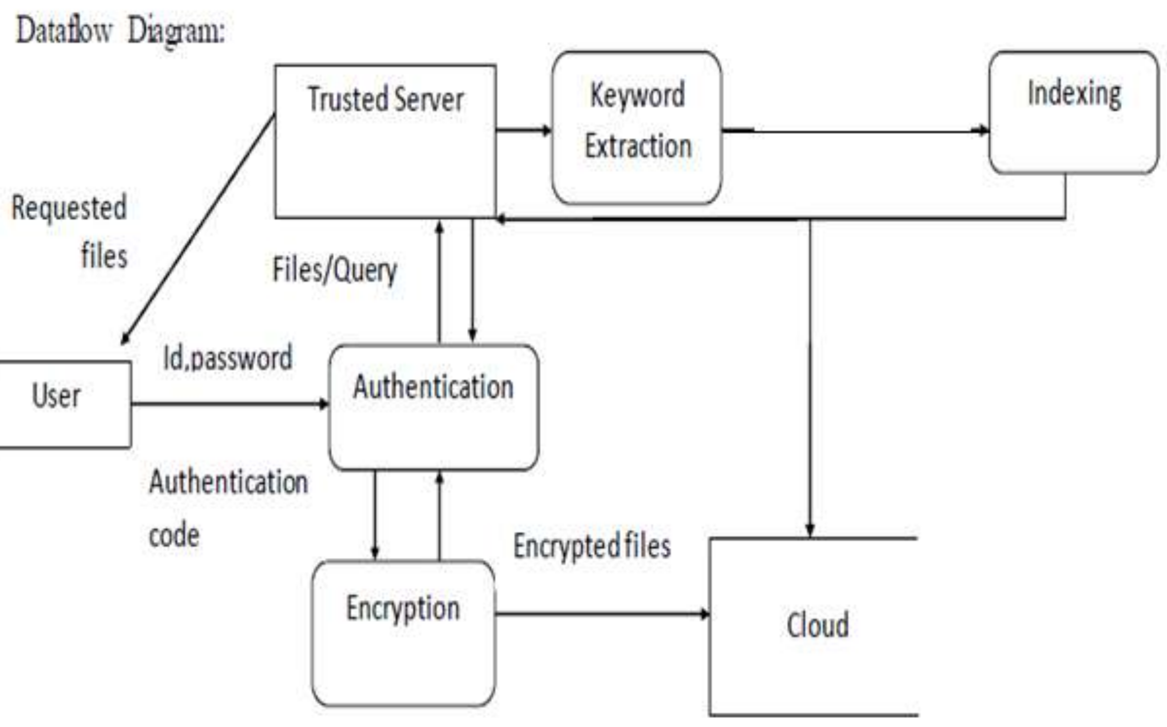


Fig. 8: Dataflow diagram of the system

4.2.5 Activity Diagram (for the system)

The following diagram depicts the activity diagram of the entire system under implementation.

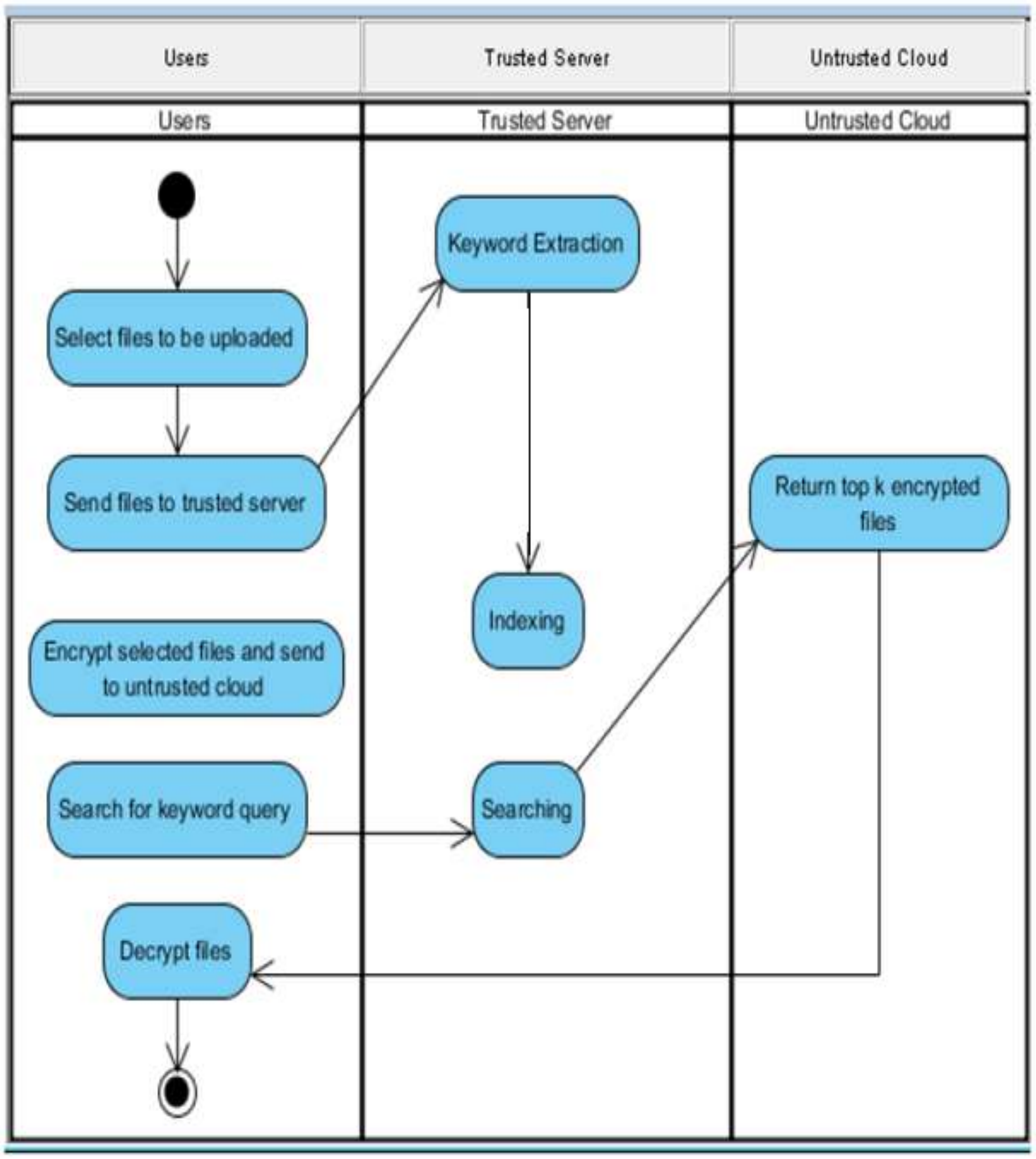


Fig. 9: Activity diagram of the system

4.2.6 Deployment Diagram (for the system)

The following diagram represents the deployment diagram of the entire system and indicates its functioning.

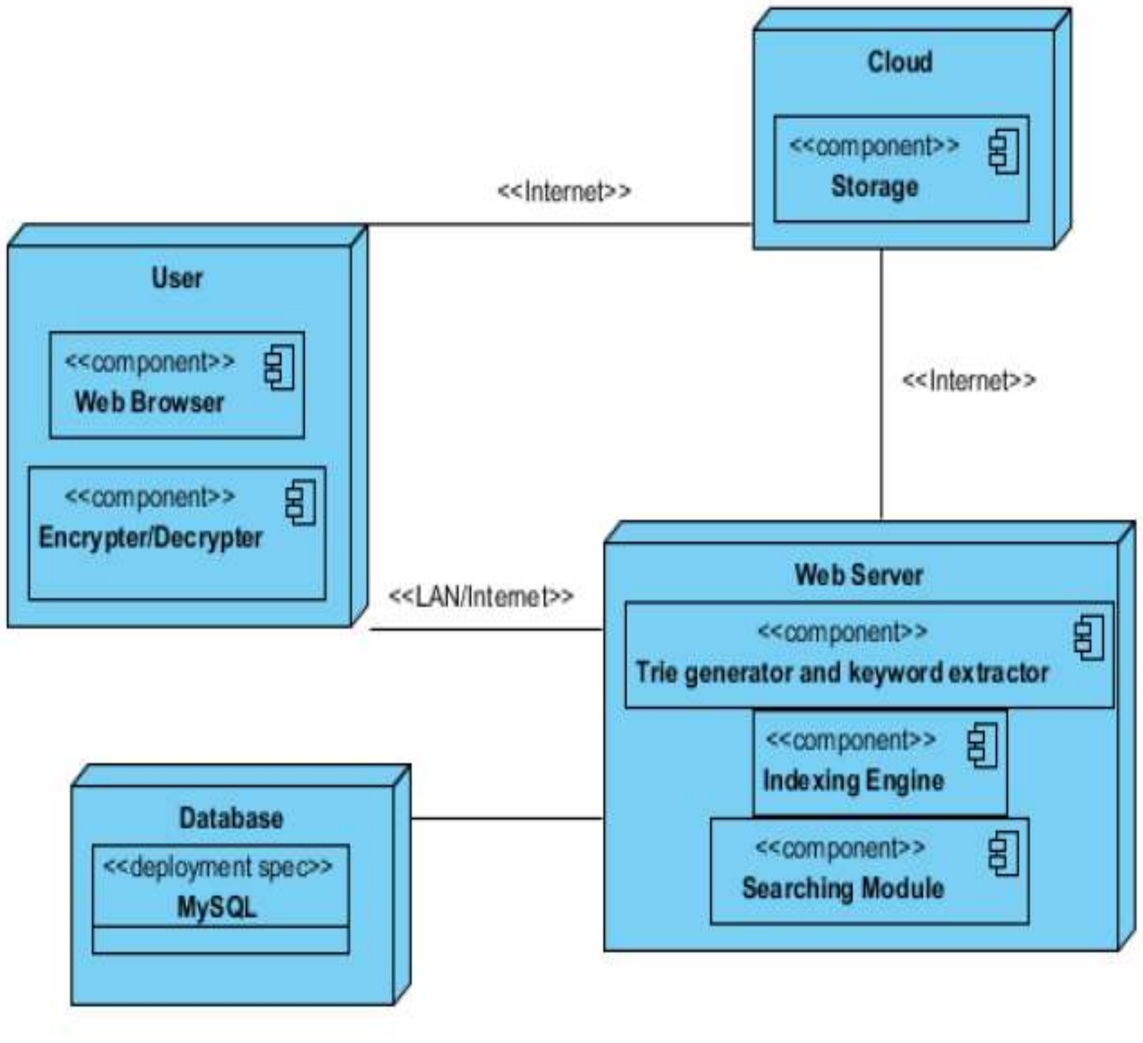


Fig. 10: Deployment diagram of the system

4.2.7 Flowchart:

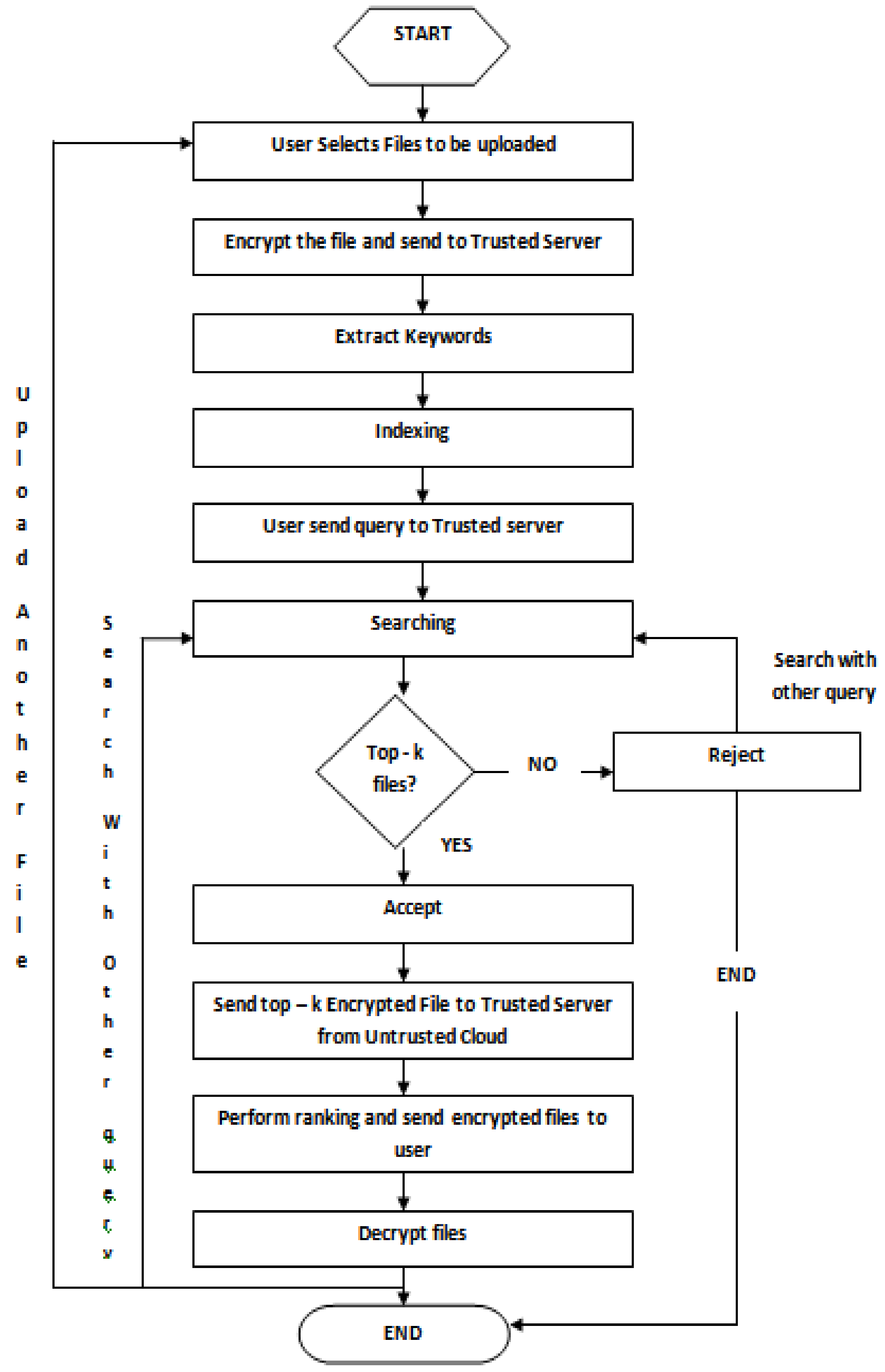

START
User Selects Files to be uploaded
Encrypt the file and send to Trusted Server
Extract Keywords
Indexing
User send query to Trusted server
Searching
Top - k files?
NO
Reject
Search with other query
YES
Accept
Send top – k Encrypted File to Trusted Server from Untrusted Cloud
Perform ranking and send encrypted files to user
Decrypt files
END
END
Upload Another File
Search With Other query

Fig. 11: Flowchart for the system

4.3 PLAN OF WORK

Sr. No	Activity	Start Time	Duration	End Time
1	Analysis of Project Definition	0	15	15
2	Finalizing the Problem Definition	15	2	17
3	Literature and statistical analysis and survey	17	30	47
4	Analysis of previous work	42	7	49
5	Design of our implementation	49	5	54
6	Analysis of our design	54	2	56
7	Creating a prototype of our design	56	5	61
8	Creating UI of Application	61	3	64
9	Debugging and Building	64	2	66
10	Implementing searching techniques using examples	66	8	74
11	Exception and Runtime Testing	74	2	76
12	Simulation of partial project	76	2	78
13	Implementing additional functionalities to project	78	4	82
14	Calculating and comparing space and time complexities	82	5	87
15	Testing the final project	87	5	92
16	Simulation of final project	92	3	95
17	Review	95	4	99
18	Release	99	1	100

Table 1: Plan of work

4.4 PROJECT SCHEDULING & TRACKING USING TIME LINE / GANTT CHART

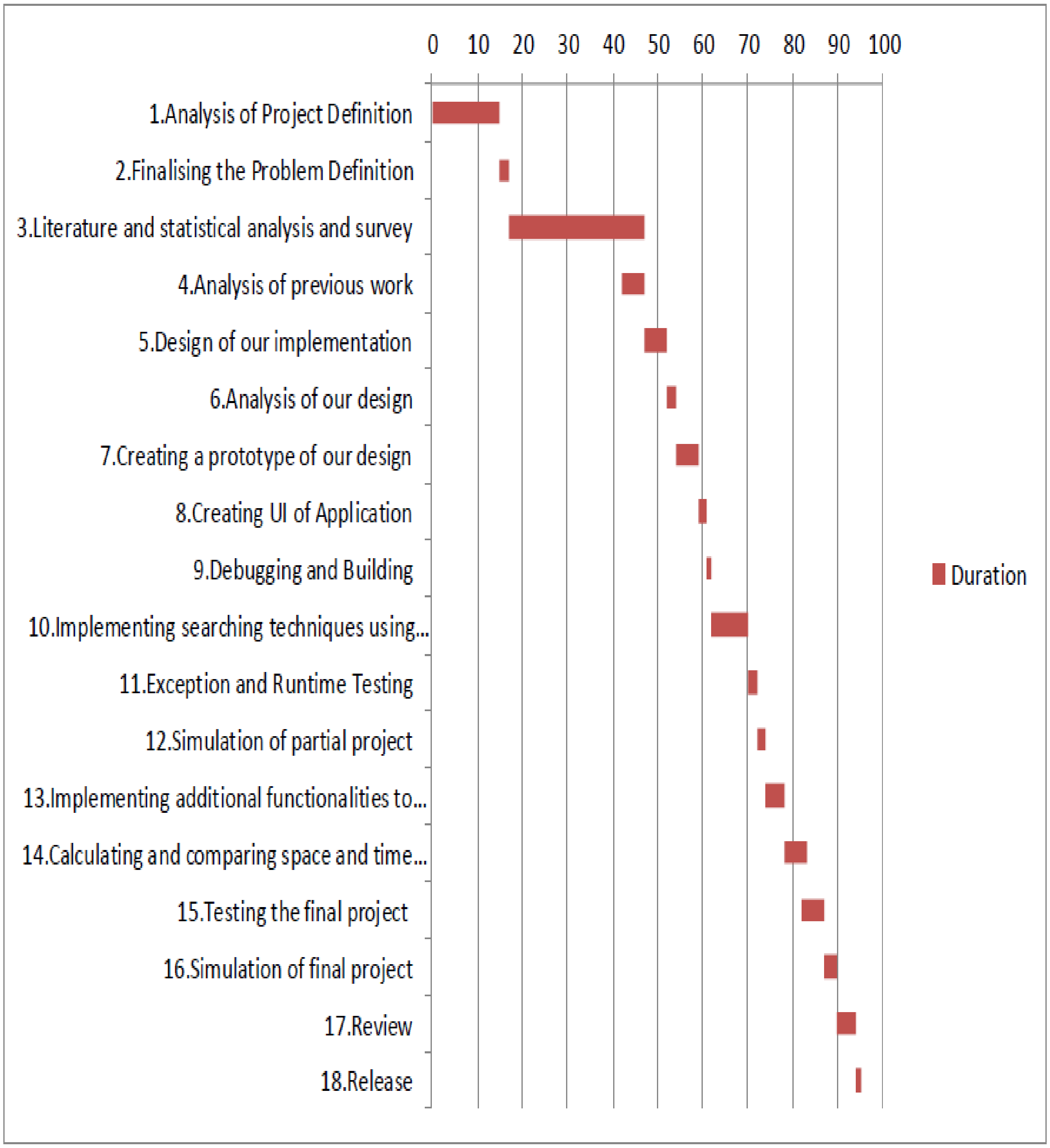


Fig 12: Gantt chart

5. IMPLEMENTATION

5.1 AES ENCRYPTION ALGORITHM

As we are not going to perform any operation on the outsourced files to search of the keywords, we can use any of the existing light weight symmetric key Encryption algorithms and unload the data files to the cloud. We use AES to encrypt the file and then outsource it.

The overall structure of AES encryption/decryption is shown in figure below.

1. The number of rounds shown in figure is for the case when the encryption key is 128 bit long.
2. Before any round-based processing for encryption can begin, the input state array is XORed with the first four words of the key schedule. The same thing happens during decryption — except that now we XOR the cipher text state array with the last four words of the key schedule.
3. For encryption, each round consists of the following four steps: 1) Substitute bytes, 2) Shift rows, 3) Mix columns, and 4) Add round key. The last step consists of XORing the output of the previous three steps with four words from the key schedule.
4. For decryption, each round consists of the following four steps: 1) Inverse shift rows, 2) Inverse substitute bytes, 3) Add round key, and 4) Inverse mix columns. The third step consists of XORing the output of the previous two steps with four words from the key schedule. Note the difference between the order in which substitution and shifting operations are carried out in a decryption round via-à via the order in which similar operations are carried out in an encryption round.
5. The last round for encryption does not involve the “Mix columns” step. The last round for decryption does not involve the “Inverse mix columns” step.

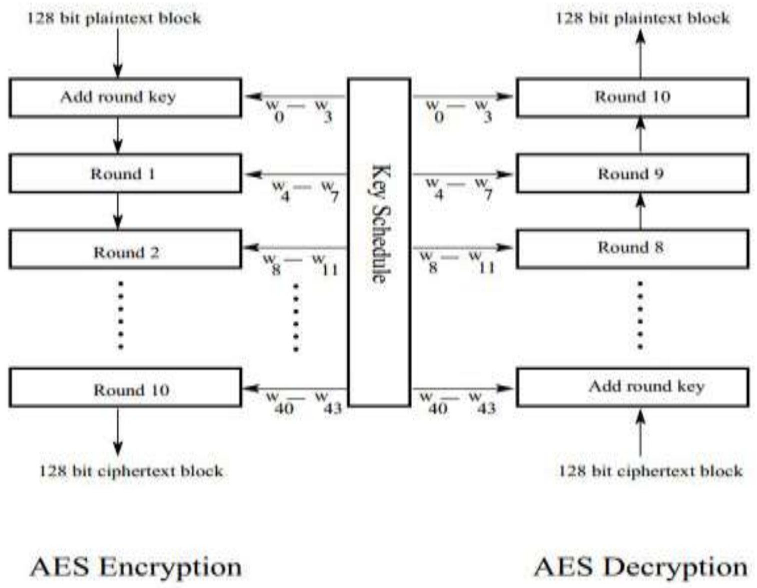


Fig. 13: AES Block Diagram

5.1.1 Reason for selecting AES Algorithm.

1. The cipher AES-256 is used among other places in TSL/SSL across the Internet. It's considered among the top ciphers.
2. In theory it's not crackable since the combinations of keys are massive.
3. Advanced Encryption Standard not only assures security but also improves the performance in a variety of settings such as smartcards, hardware implementations etc.
4. AES is federal information processing standard and there are currently no known non-brute-force direct attacks against AES.
5. AES is strong enough to be certified for use by the US government for top secret information.

5.2 KEYWORD GENERATION

1. Accept the string (contents of files) as an argument.

2. Remove unwanted characters:
   a. Strip punctuation characters
   b. Strip symbol characters
   c. Strip number characters
   d. Convert to lower case

3. Process the page's words
   a. Split the text into a word list
   b. Stem the words
   c. Remove stop words (This is a list of words that occur quite a bit in English text and would therefore interfere with the outcome of the function.)
   d. Remove unwanted words
   e. Count keyword usage (frequency count of each keyword)

4. The function returns the most commonly occurring words as an array, with the key as the word and the amount of times it occurs as the value.

5.3 INDEXING

The extracted encrypted keywords are stored to database tables onto trusted server. The structure of keyword storage table and file storage table and its associated data is as follows:

File_id	**Keyword**	**Frequency**	**Term_frequency**

Table 2: Keyword Indexing

File_id	**File_name**	**File_size**

Table 3: File Indexing

5.4 SEARCHING

1. Accept the multi-keyword query from the user.

2. Encrypt the query.

3. Extract the keywords using keyword extraction algorithm described above.

4. Match the keyword occurrences: loop through all keywords and check if they match any of the fields.

5. For every field that matches with our search term, give it a score, which will be used for ranking.

6. The files would be retrieved from untrusted cloud, based on the results derived from above steps.

7. The top-k files will be selected based on ranking function implemented by trusted server.

8. Decrypted files will be sent to the user.

5.5 RANKING

We make use of Term Frequency- Inverse Document Frequency (tf-idf) to compute ranking for most relevant files.

1. Calculate the term frequency. TF (t) = (Number of times term t appears in a document) / (Total number of terms in the document).

2. Calculate the Inverse Document Frequency. IDF (t) = log_e (Total number of documents / Number of documents with term t in it).

3. Calculate TF-IDF = TF (t) * IDF (t).

5.6 DROPBOX API CONNECTION WITH PHP

1. First register a new app on the App Console. You'll need the app key to access the Core API.

2. Authenticating your app:

The Core API uses OAuth v2. You'll need to provide your app key and secret to the new DropboxOAuth.

Now we're all set to start the OAuth flow. The OAuth flow has two parts:

a. Ask the user to authorize linking your app to their Dropbox account.
b. Once authorized, exchange the received authorization code for an access token, which will be used for calling the Core API.

With the authorization URL in hand, we can now ask the user to authorize your app.

Once the user has delivered the authorization code to our app, we can exchange that code for an access token

The access token is all you'll need to make API requests on behalf of this user, so you should store it away for safe-keeping

5.7 STEPS FOR IMPLEMENTATION

1. The user first registers with the system and login action is performed.

2. The User then selects a plain text document/multimedia/database files to upload. These files are then encrypted using **AES 128 bit** algorithm.

3. Following actions are performed at Trusted Server after it receives the encrypted file.

   a. Keyword Extraction is performed
   b. Encrypted index table is generated and maintained.
   c. Encrypted file is uploaded to Dropbox via the Dropbox Api.

4. Each time the user uploads a file to Dropbox an **access token** is provided which is necessary to download the file.

5. To perform a search operation the user enters a multi keyword query which is then encrypted using **AES 128 algorithm** and **keyword extraction** is then performed.

6. Now these extracted keywords are matched against all the keywords in the database on the trusted server using **pattern matching algorithm** every matching keyword is then assigned a score used for ranking.

7. Based on these scores the **top-k** ranked files are determined and returned to the user by the untrusted cloud.

8. The user then selects the file to download based on ranking.

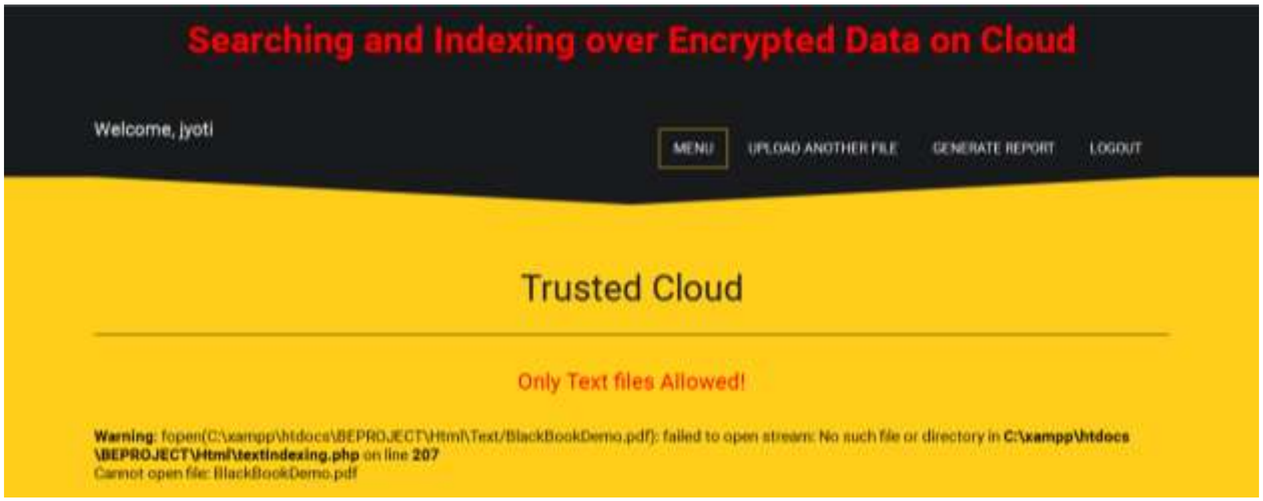
9. The access token provided earlier must be entered by the user to download the document.

10. Only if a correct access token is provided the files are downloaded.

11. The user then decrypts those files.

6. TESTING

1. Incorrect type of file being uploaded: pdfs, ppts and word files cannot be uploaded.

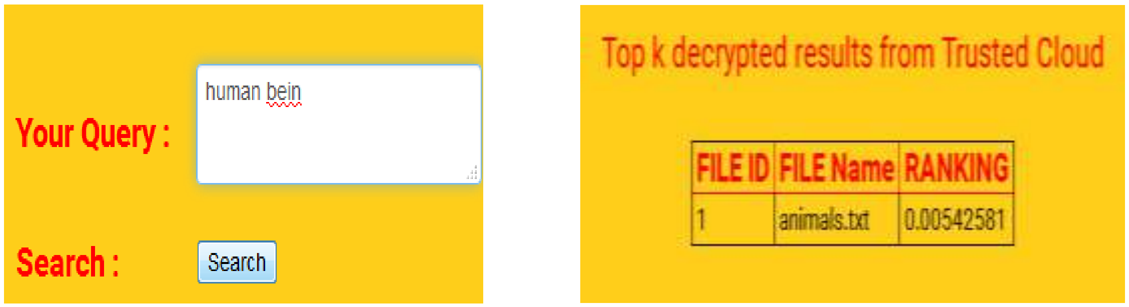


Only Text files Allowed!

```
Warning: fopen(C:\xampp\htdocs\BEPROJECT\Html\Text/BlackBookDemo.pdf): failed to open stream: No such file or directory in C:\xampp\htdocs
\BEPROJECT\Html\textindexing.php on line 207
Cannot open file: BlackBookDemo.pdf
```

Fig. 14: Incorrect type of file being uploaded.

2. Function for searching relevant files even when keyword entered is wrong or misspelled.



Your Query : human bein

FILE ID	FILE Name	RANKING
1	animals.txt	0.00542581

Fig. 15: Searching relevant files when keyword entered is wrong or misspelled.

3. Function that allows re-entering correct access token: If wrong access token is entered user is allowed to download and enter it again.

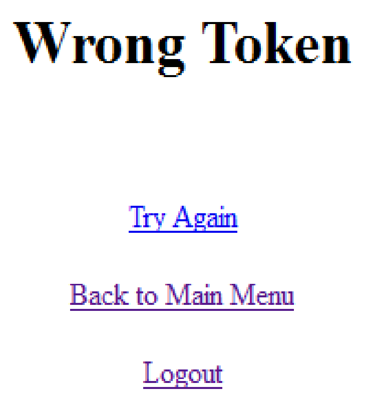
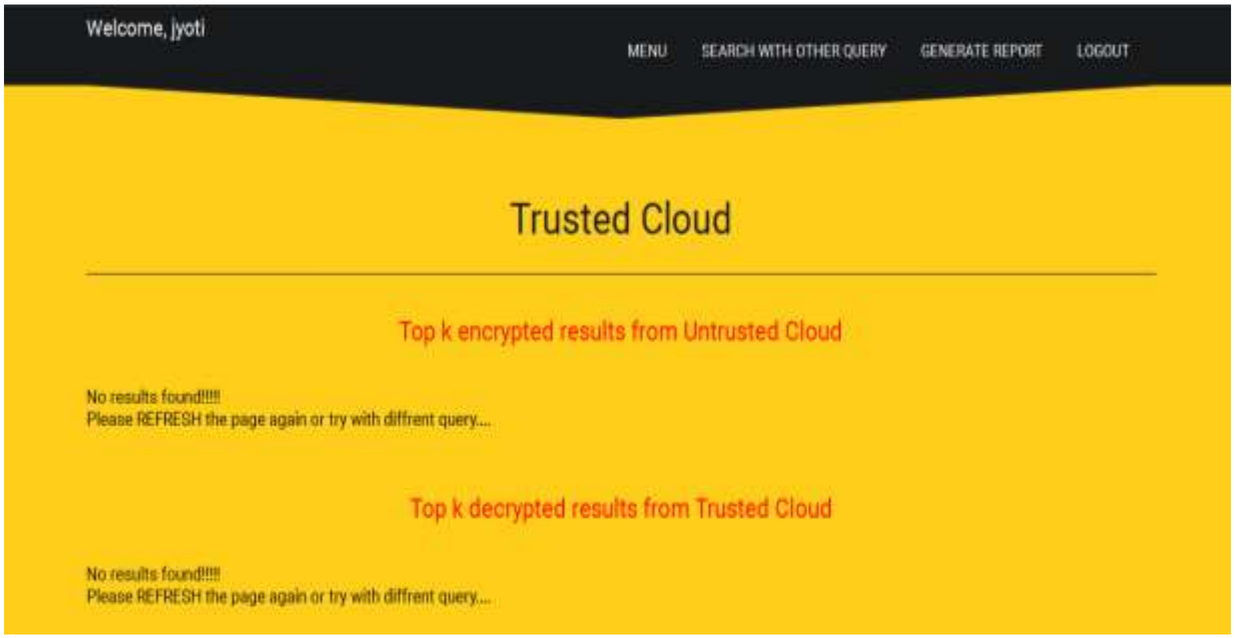

Wrong Token

Fig. 16: Re-entering Access Token.

4. Files with no relevant keywords can be uploaded but cannot be indexed:

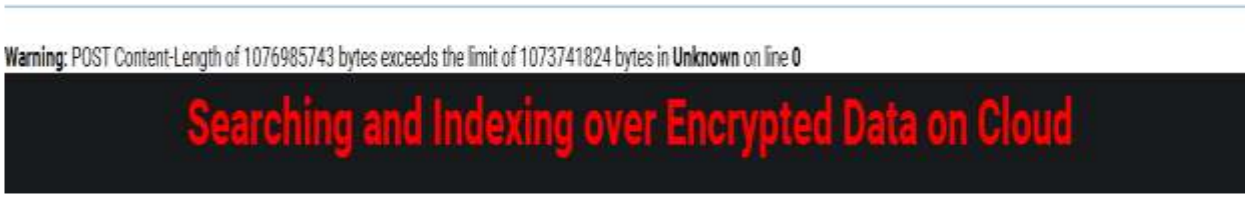

Welcome, jyoti
MENU
SEARCH WITH OTHER QUERY
GENERATE REPORT
LOGOUT
Trusted Cloud
Top k encrypted results from Untrusted Cloud
Top k decrypted results from Trusted Cloud

No results found!!!!
Please REFRESH the page again or try with diffrent query....

No results found!!!!
Please REFRESH the page again or try with diffrent query....

Fig. 17: No results found error!

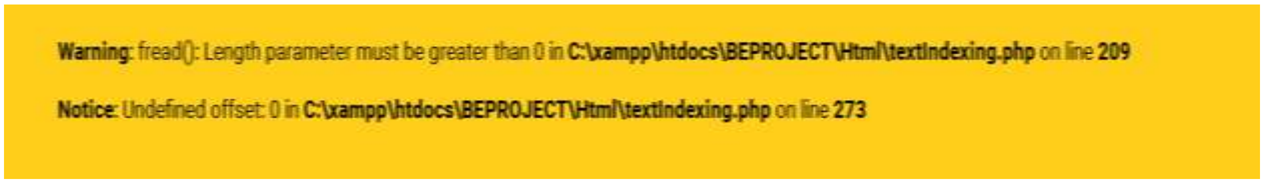
5. Multimedia files cannot exceed size by 1024 MB .i.e. maximum size of file that can be uploaded is 1024 MB:



```
Warning: POST Content-Length of 1076985743 bytes exceeds the limit of 1073741824 bytes in Unknown on line 0
```

Fig. 18: POST Content-Length error.

6. Files with minimum size of 0 bytes can be uploaded but cannot be indexed.

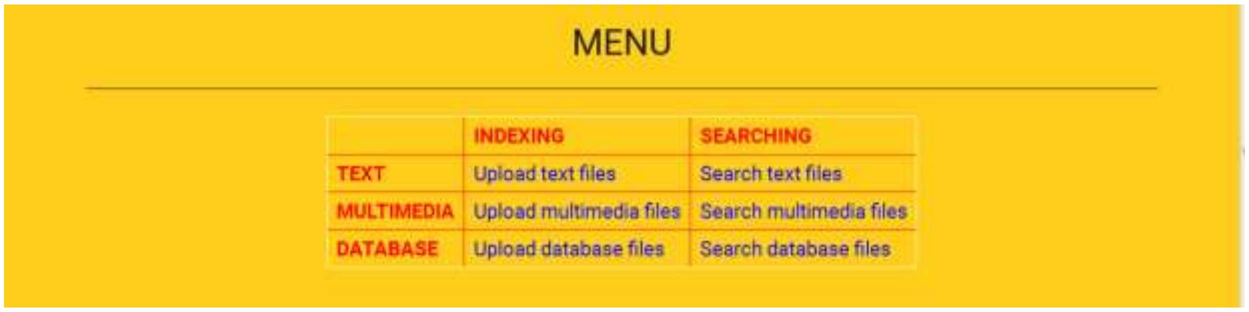


```
Warning: fread(): Length parameter must be greater than 0 in C:\xampp\htdocs\BEPROJECT\Html\textIndexing.php on line 209

Notice: Undefined offset: 0 in C:\xampp\htdocs\BEPROJECT\Html\textIndexing.php on line 273
```

Fig. 19: Zero length error.

7. User can download any file after searching without uploading any other file in same session and also vice versa.

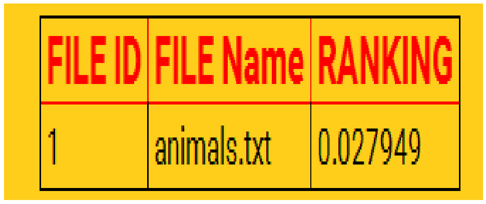


MENU

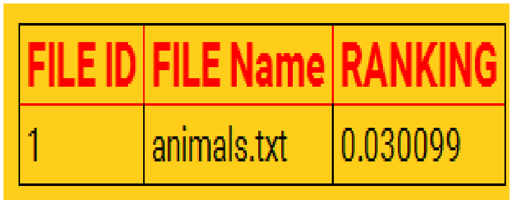
	INDEXING	SEARCHING
TEXT	Upload text files	Search text files
MULTIMEDIA	Upload multimedia files	Search multimedia files
DATABASE	Upload database files	Search database files

Fig. 20: Independent paths for uploading, searching and downloading.

8. When same file in retrieved by different keywords in query its rank is added.



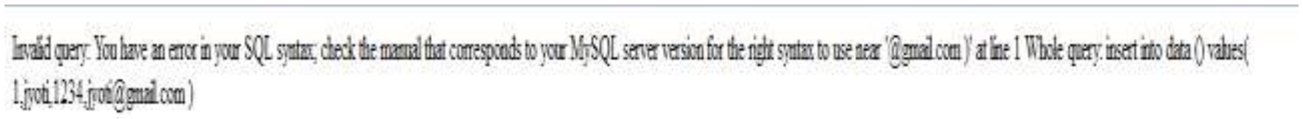
FILE ID	FILE Name	RANKING
1	animals.txt	0.027949



FILE ID	FILE Name	RANKING
1	animals.txt	0.030099

Fig. 21: Loops in operational process.

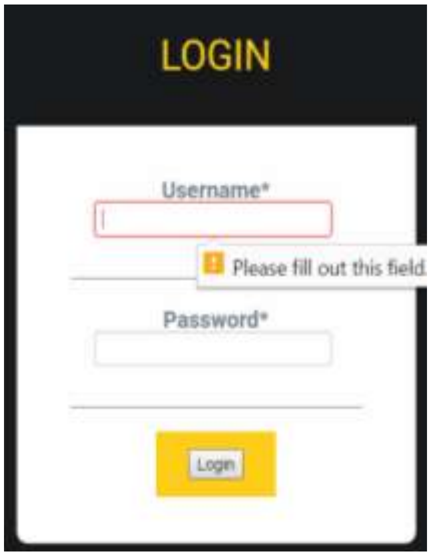
9. Database file with mismatching column data cannot be uploaded:



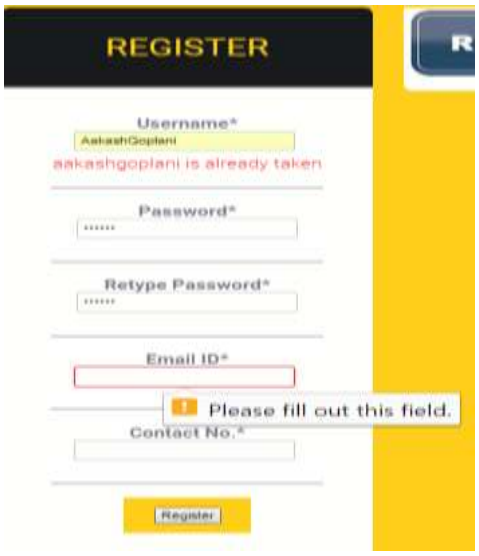
```
Invalid query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '@gmail.com )' at line 1 Whole query: insert into data () values(
1,jyoti,1234,jyoti@gmail.com )
```

Fig. 22: Internal data structure is available for validation.

10. Fields cannot be left blank in any of form fields.



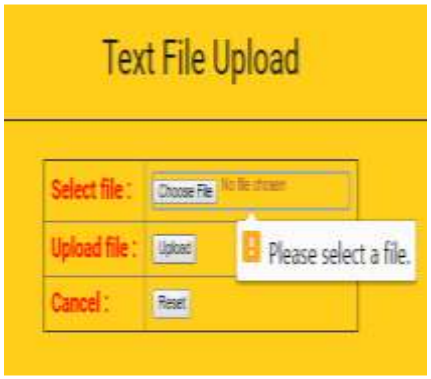
Please fill out this field.



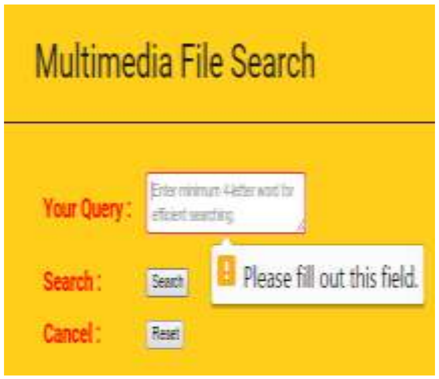
AakashGoplani

aakashgoplani is already taken

Please fill out this field.



Please select a file.



Enter minimum 4-letter word for efficient searching.

Please fill out this field.

Fig. 23: Empty text field error.

7. RESULT ANALYSIS

7.1 SIMULATION MODEL

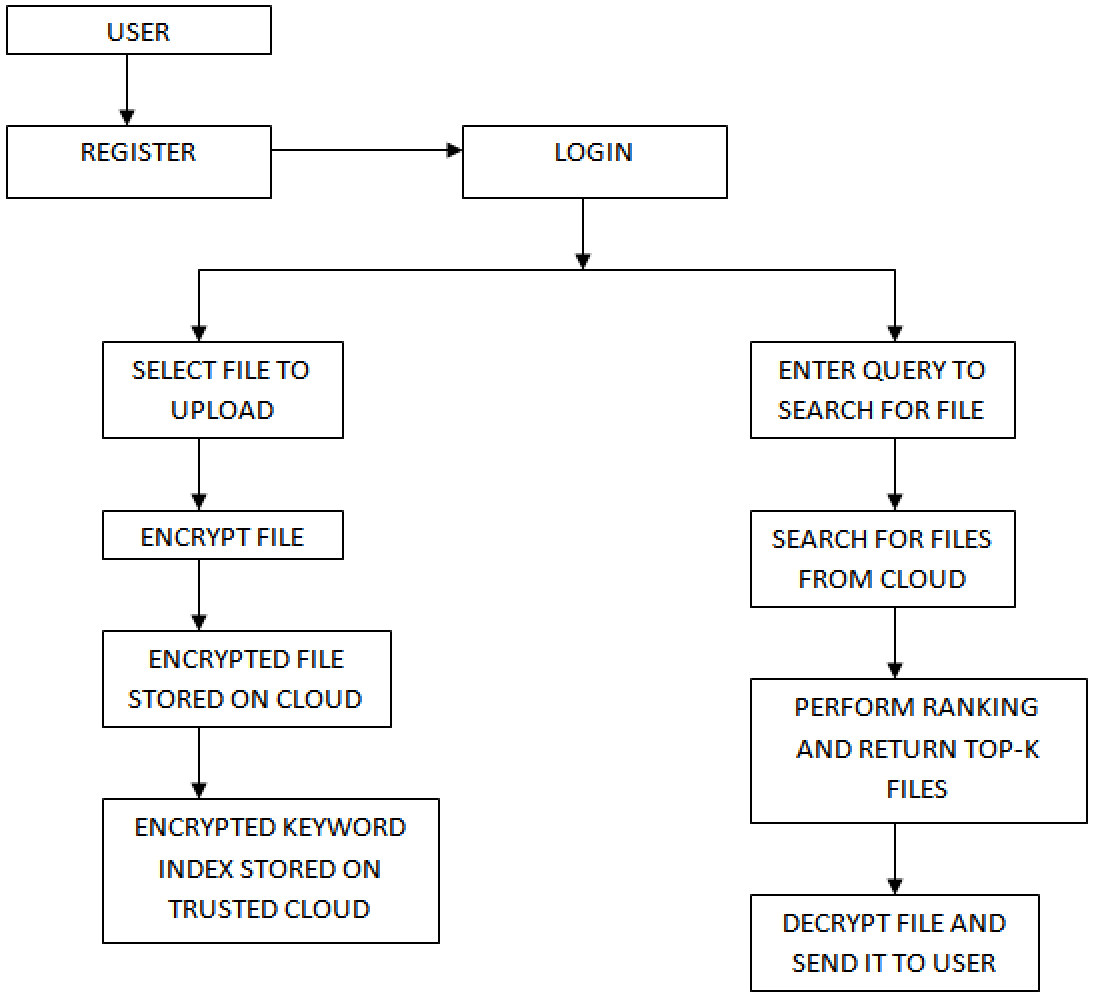
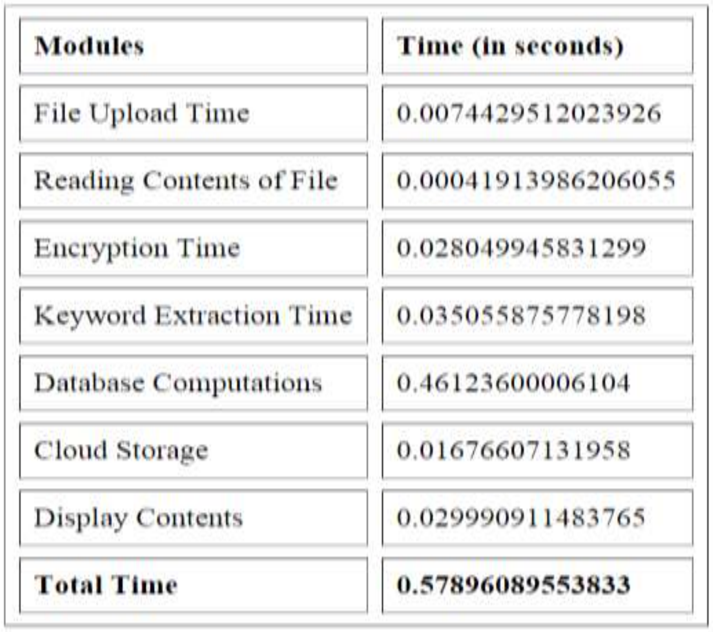

USER
REGISTER
LOGIN
SELECT FILE TO UPLOAD
ENCRYPT FILE
ENCRYPTED FILE STORED ON CLOUD
ENCRYPTED KEYWORD INDEX STORED ON TRUSTED CLOUD
ENTER QUERY TO SEARCH FOR FILE
SEARCH FOR FILES FROM CLOUD
PERFORM RANKING AND RETURN TOP-K FILES
DECRYPT FILE AND SEND IT TO USER

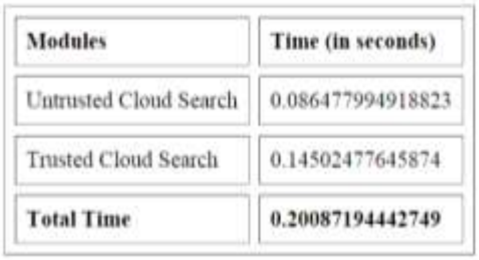
Fig. 24: Simulation Model.

7.2 PARAMETERS / GRAPHS



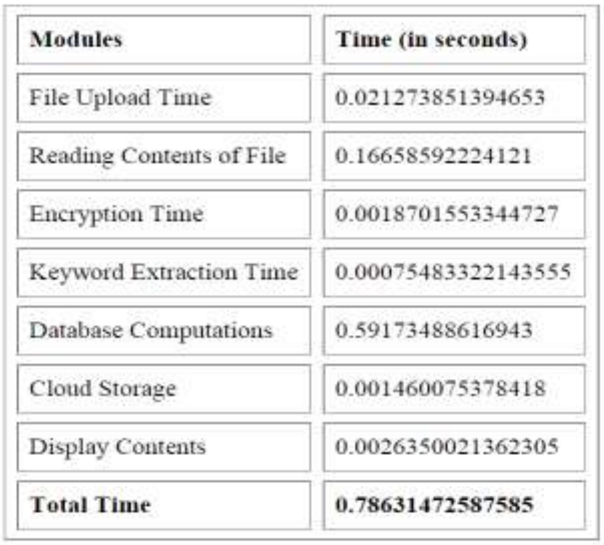
Modules	**Time (in seconds)**
File Upload Time	0.0074429512023926
Reading Contents of File	0.00041913986206055
Encryption Time	0.028049945831299
Keyword Extraction Time	0.035055875778198
Database Computations	0.46123600006104
Cloud Storage	0.01676607131958
Display Contents	0.029990911483765
Total Time	**0.57896089553833**

Fig. 25: Time Analysis – Encrypted Text Indexing.



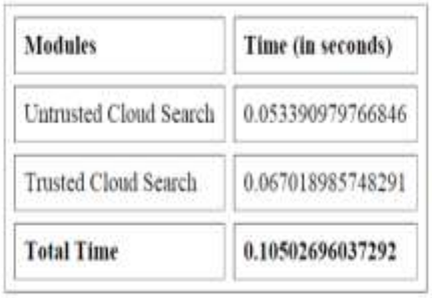
Modules	**Time (in seconds)**
Untrusted Cloud Search	0.086477994918823
Trusted Cloud Search	0.14502477645874
Total Time	**0.20087194442749**

Fig. 26: Time Analysis – Encrypted Text Searching.



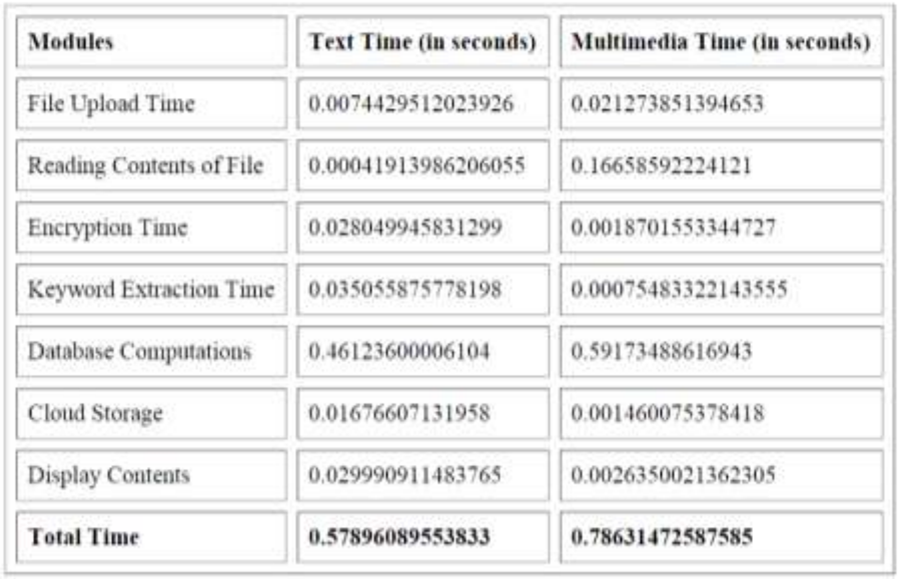
Modules	**Time (in seconds)**
File Upload Time	0.021273851394653
Reading Contents of File	0.16658592224121
Encryption Time	0.0018701553344727
Keyword Extraction Time	0.00075483322143555
Database Computations	0.59173488616943
Cloud Storage	0.001460075378418
Display Contents	0.0026350021362305
Total Time	**0.78631472587585**

Fig. 27: Time Analysis – Encrypted Multimedia Indexing.



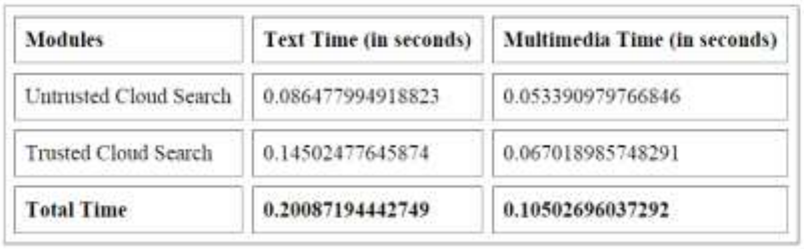
Modules	Time (in seconds)
Untrusted Cloud Search	0.053390979766846
Trusted Cloud Search	0.067018985748291
Total Time	**0.10502696037292**

Fig. 28: Time Analysis – Encrypted Multimedia Searching.



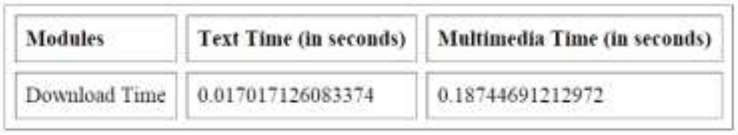
Modules	Text Time (in seconds)	Multimedia Time (in seconds)
File Upload Time	0.0074429512023926	0.021273851394653
Reading Contents of File	0.00041913986206055	0.16658592224121
Encryption Time	0.028049945831299	0.0018701553344727
Keyword Extraction Time	0.035055875778198	0.00075483322143555
Database Computations	0.46123600006104	0.59173488616943
Cloud Storage	0.01676607131958	0.001460075378418
Display Contents	0.029990911483765	0.0026350021362305
Total Time	**0.57896089553833**	**0.78631472587585**

Fig. 29: Time Analysis – Encrypted Text vs. Multimedia Indexing.



Modules	Text Time (in seconds)	Multimedia Time (in seconds)
Untrusted Cloud Search	0.086477994918823	0.053390979766846
Trusted Cloud Search	0.14502477645874	0.067018985748291
Total Time	**0.20087194442749**	**0.10502696037292**

Time Analysis : Decrypted Text vs. Encrypted Multimedia Downloading



Modules	Text Time (in seconds)	Multimedia Time (in seconds)
Download Time	0.017017126083374	0.18744691212972

Fig. 30: Time Analysis – Encrypted Text vs. Multimedia Searching and Downloading.

7.3 OUTPUT PRINTOUTS

1. The user needs to be registered and sign in to upload or download the files.

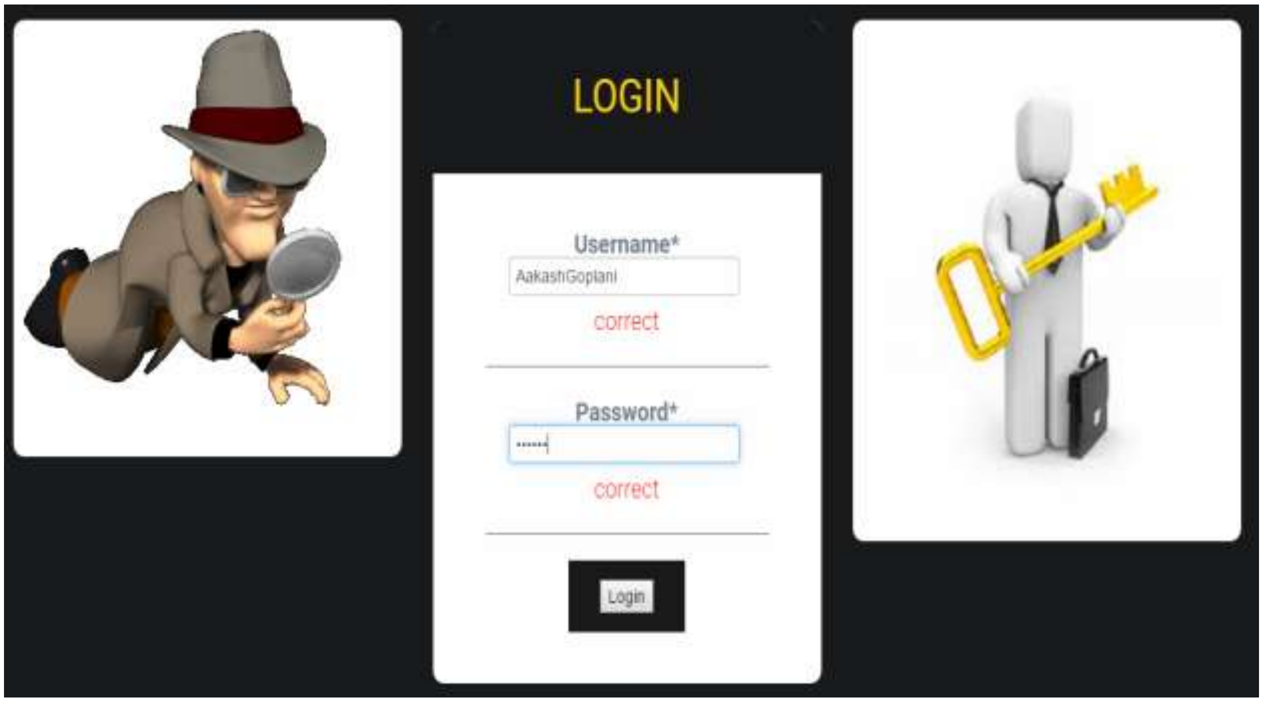
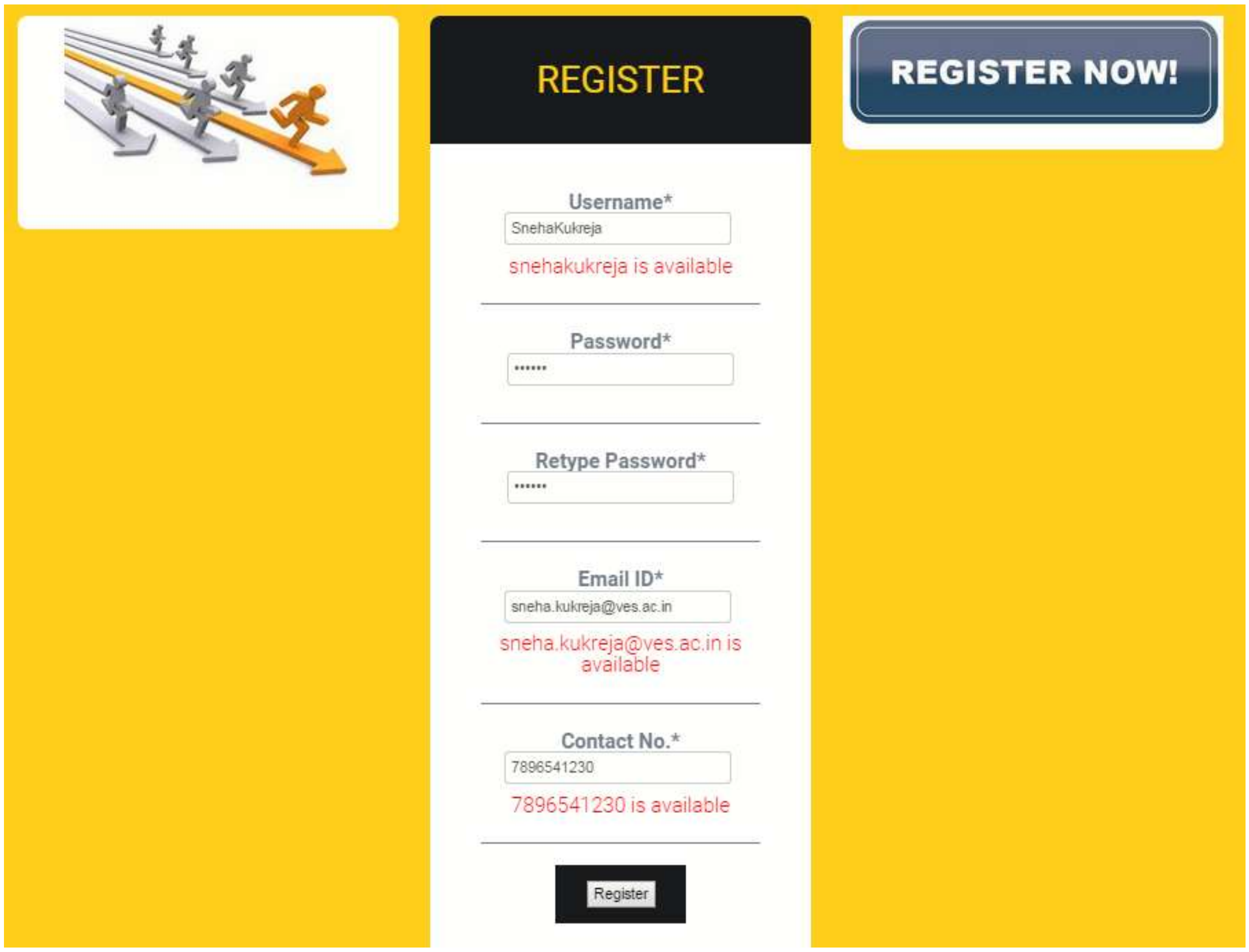


Fig. 31: Login Page.

2. User needs to register himself on the website.



REGISTER

Username*
SnehaKukreja
snehakukreja is available

Password*
••••••

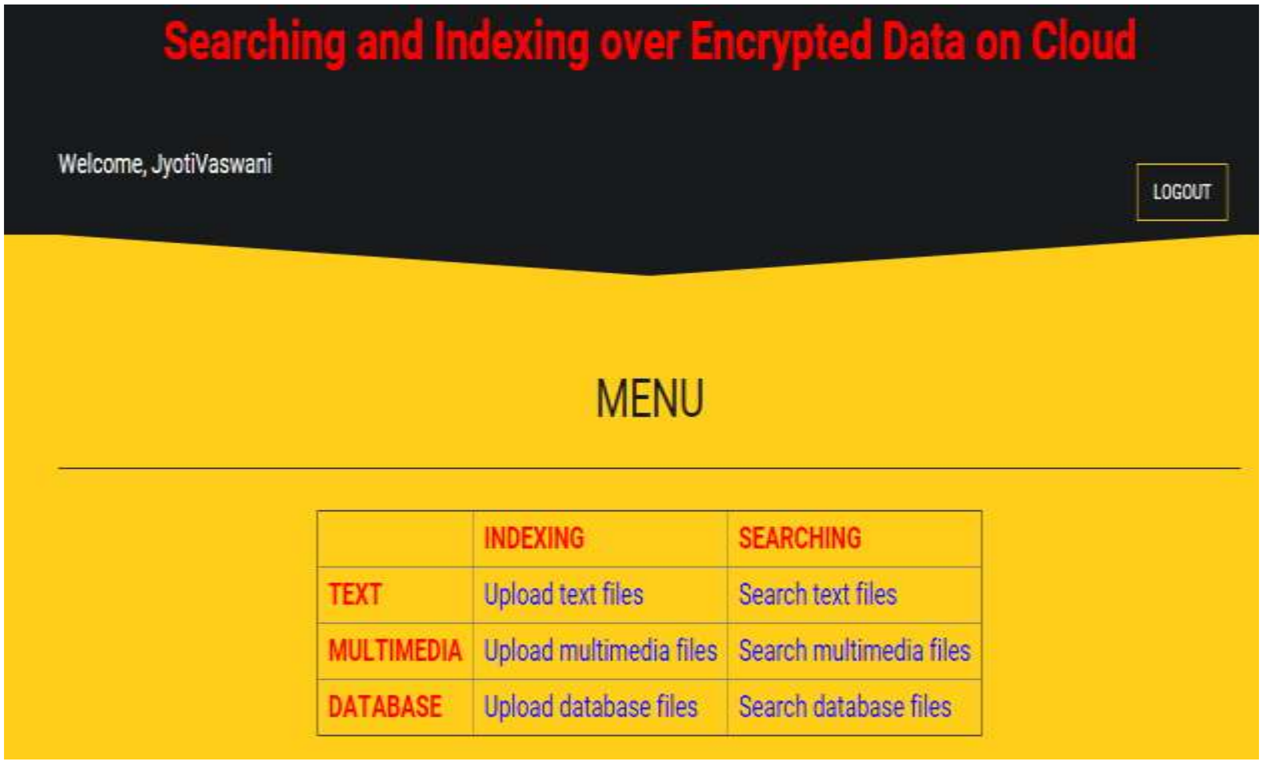
Retype Password*
••••••

Email ID*
sneha.kukreja@ves.ac.in
sneha.kukreja@ves.ac.in is available

Contact No.*
7896541230
7896541230 is available

Fig. 32: Registration Page.

3. Menu page displayed for user to select any of the options provided as shown above.



Searching and Indexing over Encrypted Data on Cloud

Welcome, JyotiVaswani

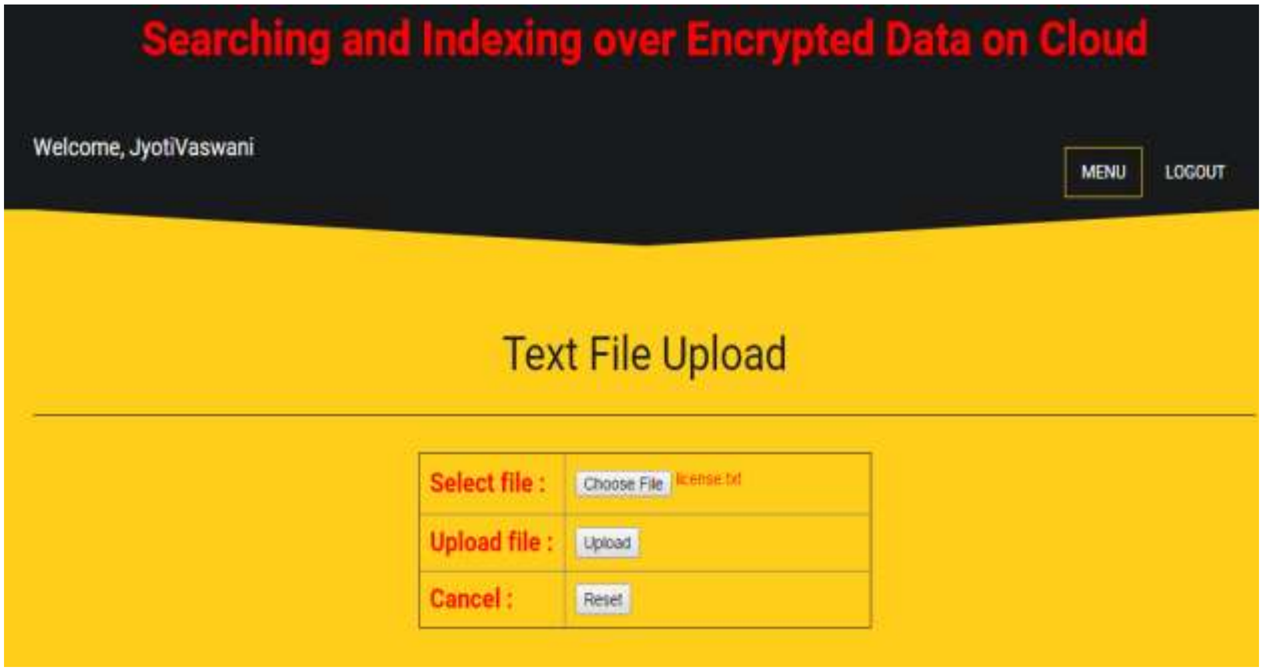
LOGOUT

MENU

	INDEXING	SEARCHING
TEXT	Upload text files	Search text files
MULTIMEDIA	Upload multimedia files	Search multimedia files
DATABASE	Upload database files	Search database files

Fig. 33: Menu page.

4. User can upload files to cloud.



Searching and Indexing over Encrypted Data on Cloud

Welcome, JyotiVaswani

MENU LOGOUT

Text File Upload

Select file :	Choose File license.txt
Upload file :	Upload
Cancel :	Reset

Fig. 34: upload a text file.

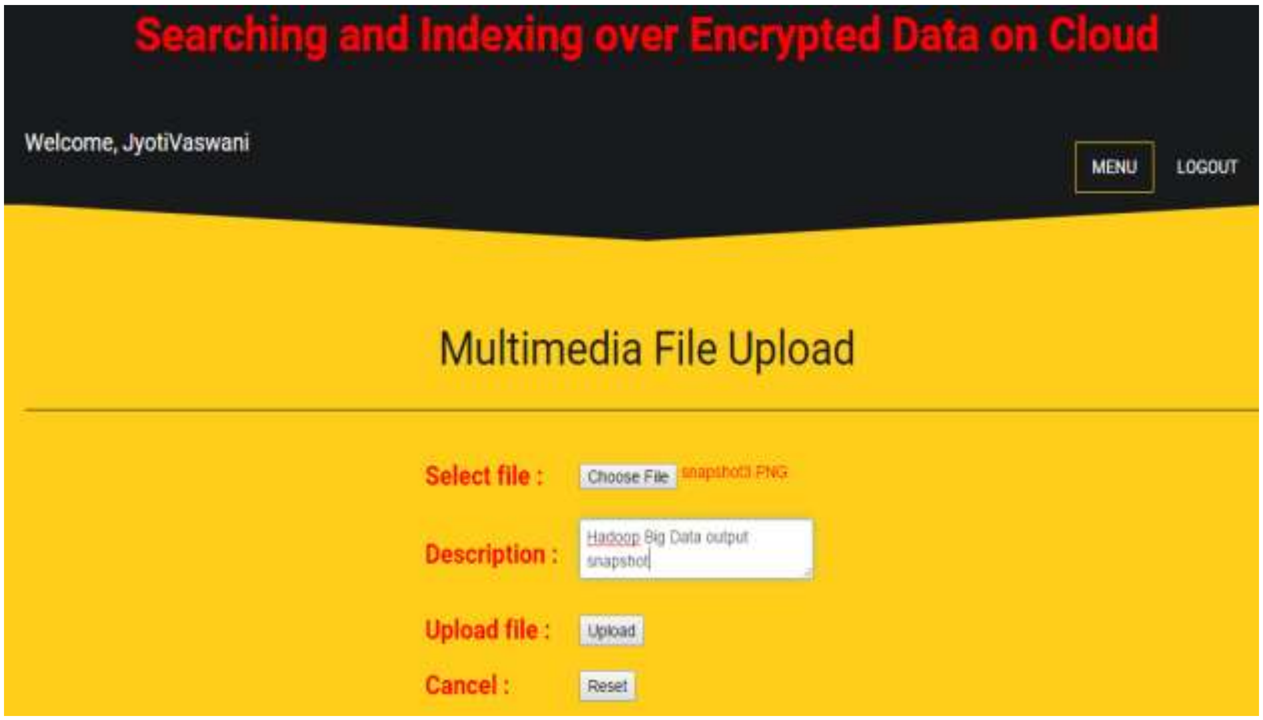


Fig. 35: upload a multimedia file.

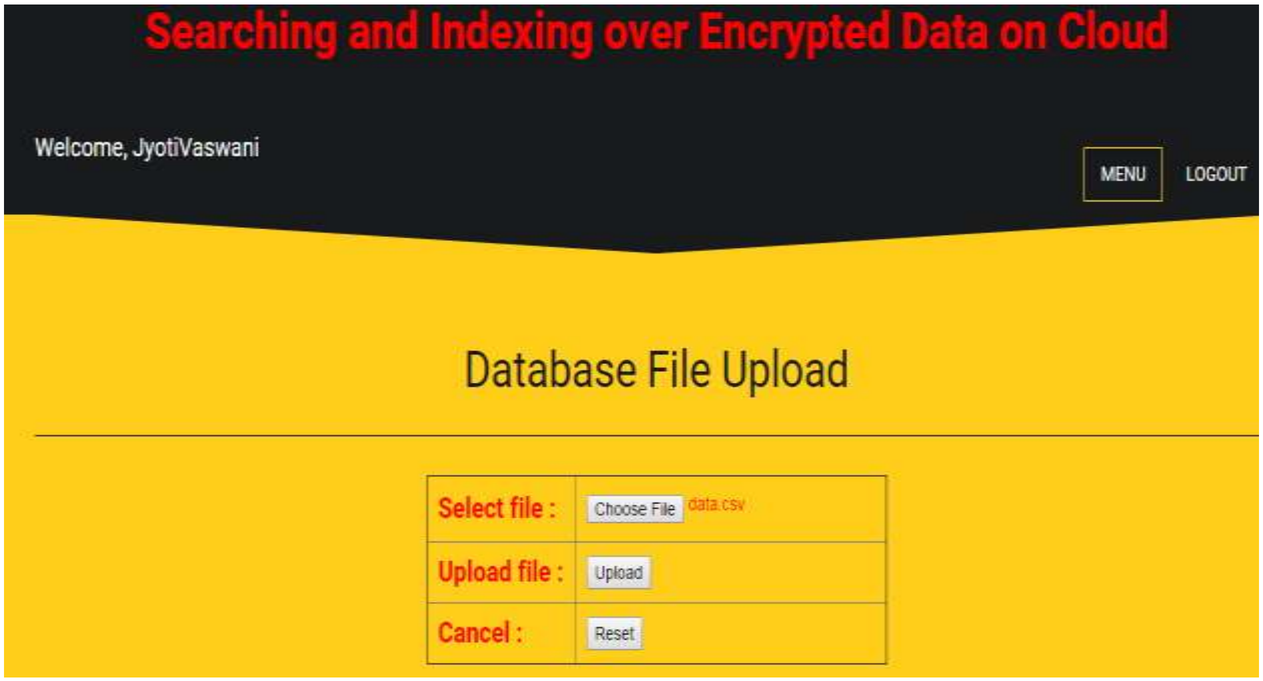
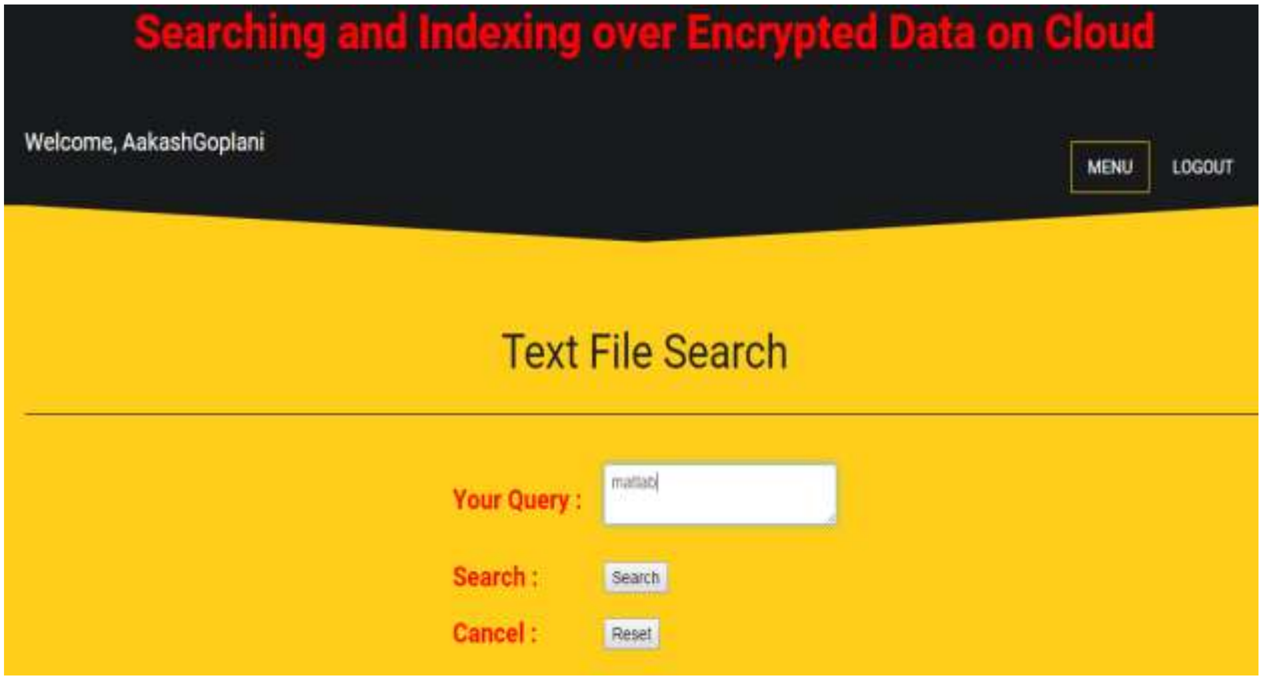


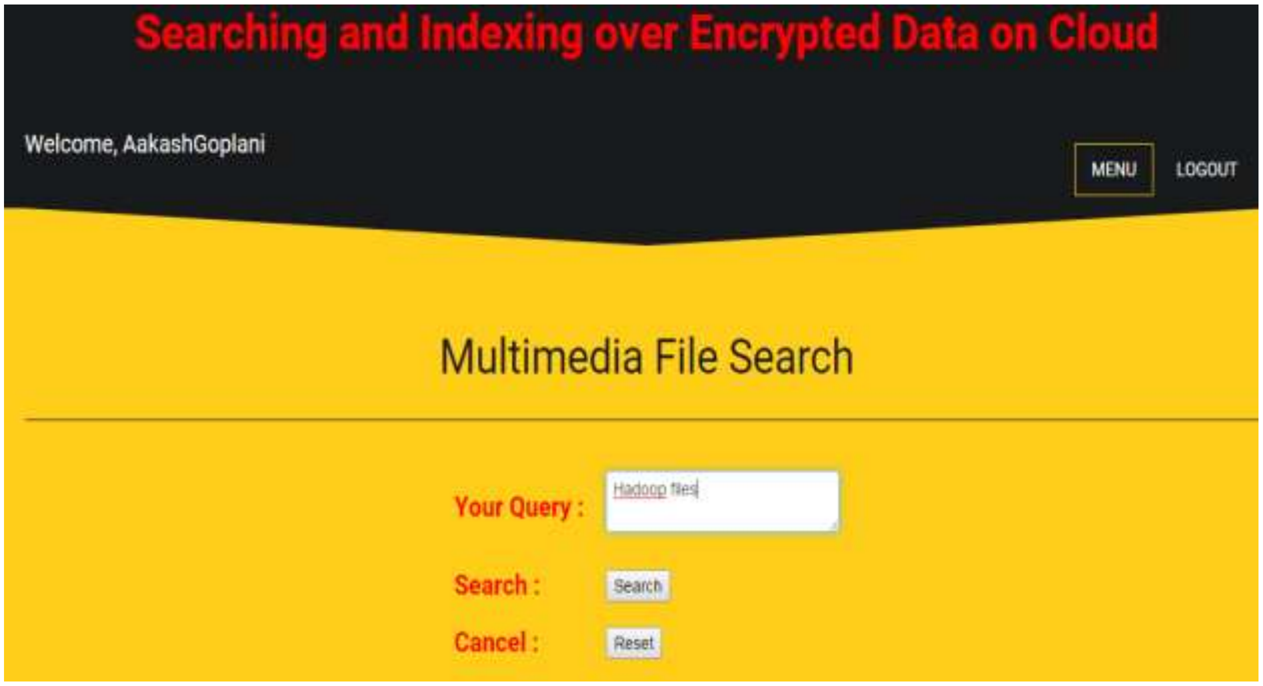
Fig. 36: upload a database file.

5. User can enter multi-keyword query and search for files.



mattab

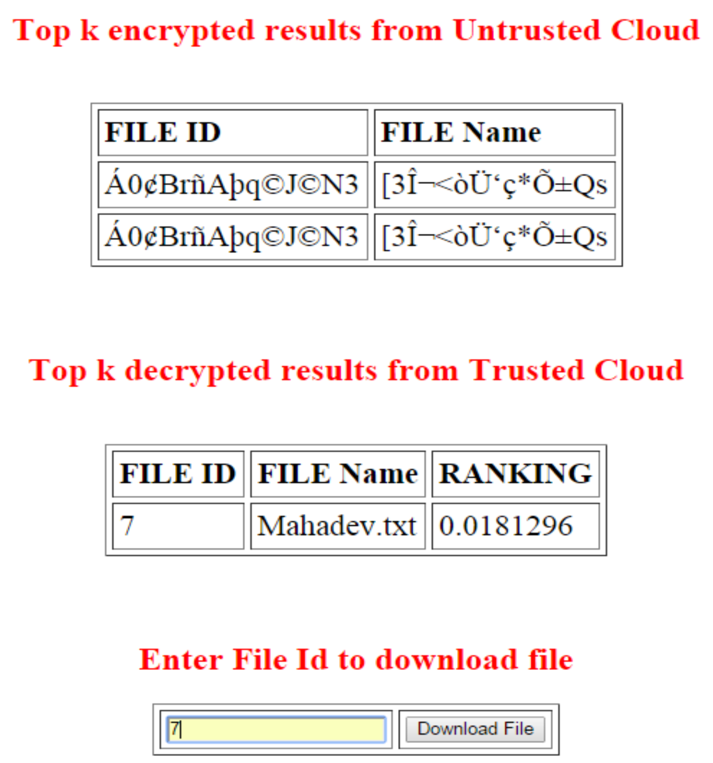
Fig. 37: Enter a query to search for a text file.



Hadoop files

Fig. 38: Enter a query to search for a multimedia file.

5. Top-k results returned from Cloud. User can enter File_id to download the decrypted file.



Top k encrypted results from Untrusted Cloud

FILE ID	FILE Name
Á0¢BrñAþq©J©N3	[3Î¬<òÜ‘ç*Õ±Qs
Á0¢BrñAþq©J©N3	[3Î¬<òÜ‘ç*Õ±Qs

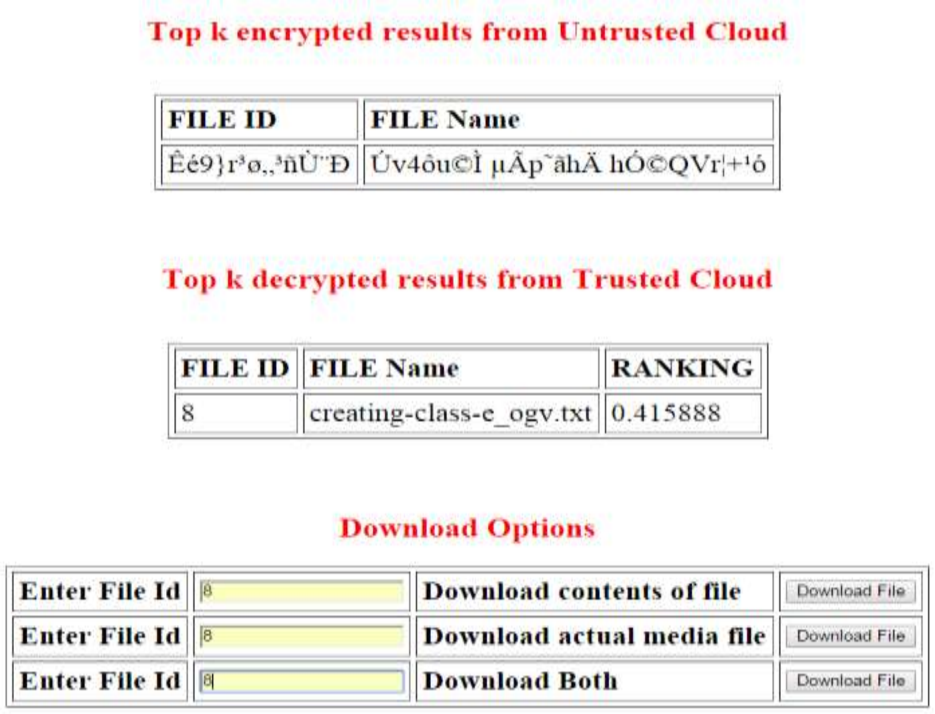
Top k decrypted results from Trusted Cloud

FILE ID	FILE Name	RANKING
7	Mahadev.txt	0.0181296

Enter File Id to download file

7	Download File

Fig. 39: Top-k files returned by cloud and trusted server.



Top k encrypted results from Untrusted Cloud

FILE ID	FILE Name
Êé9}r³ø,,³ñÙ¨Ð	Úv4ôu©Ì µÃp˜âhÄ hÓ©QVr¦+¹ó

Top k decrypted results from Trusted Cloud

FILE ID	FILE Name	RANKING
8	creating-class-e_ogv.txt	0.415888

Download Options

Enter File Id	8	Download contents of file	Download File
Enter File Id	8	Download actual media file	Download File
Enter File Id	8	Download Both	Download File

Fig. 40: Top-k files returned by cloud and trusted server.

7. User needs to enter token number before downloading the file.

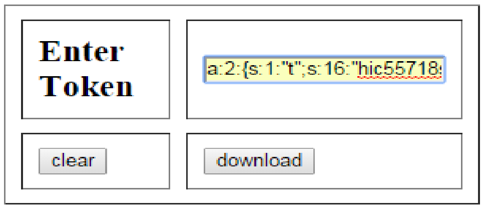


Fig. 41: Token Validation.

7.4 Observations & Analysis

Comparison of Various Techniques

Sr. No.	Scheme	Multi-keyword search	Ranked search	Fuzzy keywords search	Query based search on databases
1	Secured ranked search by Wang et al	NO	YES	NO	NO
2	Scheme by Dongxi Liu	NO	NO	NO	YES
3	Scheme by Zhangjie et al	YES	YES	NO	NO
4	Scheme by Wang et al	YES	NO	YES	NO
5	OUR SCHEME	YES	YES	PARTIAL	YES

Table 4: Observation and Analysis of Various Techniques

8. CONCLUSION

8.1 LIMITATIONS

1. Developed system only works for text data, it is unable to process pdf, ppt’s and word documents.

8.2 CONCLUSION

The main aim of the project is search over encrypted data stored on cloud without providing any information about documents or query fired by user to third party and downloading only required documents to further reduce network bandwidth.

In this project, we solve the problem of post processing overhead and unnecessary network traffic created when Boolean search techniques are used, by introducing the ranked keyword search scheme. The scheme generates indexes that help the user to search for his documents in a secure environment. The files matching the keyword search are further ranked based on the relevant score calculated with term frequency, file length etc

8.3 FUTURE SCOPE

Further extensions to the project can be done by

1. Supporting multi user environment where there would be an extra entity in the scenario i.e., data user who is authorized to access other users files. The authorization mechanisms and key exchanges methods can be modified to support the same.

2. Support for pdf, ppt and word document.

REFERENCES

Journal Paper,

[1] Dawn Xiaodong, Song David Wagner and Adrian Perrig, "Practical Techniques for Searches on Encrypted Data" in Proc. of IEEE Symposium on Security and Privacy'00, 2000.

[2] Eu–Jin Goh, "Secure indexes", in the Cryptology ePrint Archive, Report 2003/216, March 2004.

[3] Joonsang Baek, Reihaneh Safiavi- aini, Willy Susilo, "Public Key Encryption with Keyword Search Revisited", in Cryptology ePrint Archive, Report 2005/191, 2005.

[4] Hyun-A Park, Jae Hyun Park and Dong Hoon Lee, "PKIS: practical keyword index search on cloud datacenter", in EURASIP Journal on Wireless Communications and Networking 2011.

[5] Li, M., S. Yu, . Cao and W. Lou, "Authorized private keyword search over encrypted data in cloud computing." in Proceedings of the 31st International Conference on Distributed Computing Systems, June 20-24, 2011, Minneapolis, MN, USA, pp: 383-392.

[6] Ning Cao, Cong Wang, Ming Li, Kui Ren, and Wenjing Lou, "Preserving Multi-keyword Ranked Search over Encrypted Cloud Data".

[7] Steven Zittrower and Cliff C. Zou, "Encrypted Phrase Searching in the Cloud" in Globecom - Communication and Information System Security Symposium, 2012.

[8] Dongxi Liu Shenlu Wang, "Programmable Order-Preserving Secure Index for Encrypted Database Query", in IEEE Fifth International Conference on Cloud Computing, 2012.

[9] Xingming Sun, Yanling Zhu, Zhihua Xia and Liahong Chang, "Privacy- Preserving Keyword-based Semantic Search over Encrypted Cloud Data", in International Journal of Security and Its Applications Vol.8, No.3, pp.9-20, 2014.

[10] Wenhai Sun, Ahucheng Yu, Wenjing Lou and Y. Thomas Hou, "Verifiable Attribute-based Keyword Search with Fine-grained Owner-enforced Search Authorization in the Cloud", in IEEE Journal, 2013.

[11] Bing Wang, Shucheng Yu, Wenjing Lou, Y. Thomas Hou, "Privacy-Preserving Multi-Keyword Fuzzy Search over Encrypted Data in the Cloud", in IEEE I FOCOM 2014 - IEEE Conference on Computer Communications.

[12] Zhangjie Fu, Xingming Sun, igel Linge and Lu Zhou, "Multi-keyword Ranked Search over Encrypted Cloud Data Supporting Synonym Query", in IEEE Transactions on Consumer Electronics, Vol. 60, No. 1, February 2014.

[13] Xingming Sun, Lu Zhou, Zhangjie Fu and Jin Wang, "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Data in the Cloud supporting Dynamic Update", in International Journal of Security and its Application (IJSIA), Vol.8, No.6, pp.1-16, 2014.

[14] Zhihua Xia**,** Yanling Zhu**,** Xingming Sun, and Liahong Chen, "Secure Semantic expansion based search over encrypted cloud data supporting similarity ranking", in Journal of Cloud Computing, 2014.

Proceeding Paper,

[15] Ankit Doshi, Kajal Thakkar, Sahil Gupte and Anjali Yeole, “A Survey on Searching and Indexing on Encrypted Data”, in International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 2 Issue 10, October – 2013.

Patents

[1] Matzkel, et al., “System, apparatus and method for encryption and decryption of data transmitted over a network”, U.S. Patent 9,002,976, April 7, 2015.