

1. INTRODUCTION

Searchable Encryption (SE) plans give security and protection to the cloud information. The current SE approaches empower different clients to perform look activity by utilizing different plans like Broadcast Encryption, Attribute-Based Encryption, and so on. Be that as it may, these plans don't enable various clients to play out the pursuit task over the encoded information of numerous proprietors. Some SE plans include a Proxy Server (PS) that enables different clients to play out the pursuit activity. Be that as it may, these methodologies acquire gigantic computational weight on PS because of the rehashed encryption of the client questions for change reason in order to guarantee that clients' inquiry is accessible over the encoded information of different proprietors. Thus, to dispose of this computational weight on PS, this paper proposes a safe intermediary server approach that plays out the pursuit activity without changing the client questions. This methodology likewise restores the best k pertinent records to the client inquiries by utilizing Euclidean separation similitude approach. In light of the trial think about, this methodology is productive as for pursuit time and exactness.

1.1 Introduction to data science:

In this free Data Science you will have the introduction to Data Scientist roles and responsibilities, machine learning algorithms, data analysis, data manipulation, data frame, random forest, linear and logistic regression, decision trees, neural networks, Java language, Java libraries, data model, variable, set, and more. There are plenty of Data Science use cases and practical examples. Data science helps the user by providing an ability to analyze huge data sets and by doing necessary operations, data science will save precious time and makes some big profit out of it.

Description

Data science is very much popular in today's world scenario as there is a huge amount of data generated each day in different fields like mart, hospitals, colleges, etc. Users need to perform some operations by analyzing the dataset and then find something useful from that data

1.2 Data Science Process:

Step 1: Organize Data

It includes the physical storage and formatting of data and integrated finest practices in data management

Step2: Package Data

In this the prototypes are created, the visualization is built and also statistics is performed. It includes logically joining and manipulating the raw data into a new representation and package.

Step 3: Deliver Data

In this process data is delivered to those who need that data.

Data is the new Oil. This statement shows how every modern IT system is driven by capturing, storing and analyzing data for various needs. Be it about making decision for business, forecasting weather, studying protein structures in biology or designing a marketing campaign. All of these scenarios involve a multidisciplinary approach of using mathematical models, statistics, graphs, databases and of course the business or scientific logic behind the data analysis. So we need a programming language which can cater to all these diverse needs of data science. Java shines bright as one such language as it has numerous libraries and built in a feature which makes it easy to tackle the needs of Data science.

In this tutorial we will cover these various techniques used in data science using the Java programming language.

Data science is the process of deriving knowledge and insights from a huge and diverse set of data through organizing, processing and analyzing the data. It involves many different disciplines like mathematical and statistical modeling, extracting data from its source and applying data visualization techniques. Often it also involves handling big data technologies to gather both

structured and unstructured data. Below we will see some example scenarios where Data science is used..

1.2.1 Recommendation systems:

As online shopping becomes more prevalent, the e-commerce platforms are able to capture users shopping preferences as well as the performance of various products in the market. This leads to creation of recommendation systems which create models predicting the shopper's needs and show the products the shopper is most likely to buy.

1.2.2 Financial Risk management:

The financial risk involving loans and credits are better analysed by using the customers past spend habits, past defaults, other financial commitments and many socio-economic indicators. These data is gathered from various sources in different formats. Organizing them together and getting insight into customers profile needs the help of Data science. The outcome is minimizing loss for the financial organization by avoiding bad debt.

1.2.3 Improvement in Health Care services:

The health care industry deals with a variety of data which can be classified into technical data, financial data, patient information, drug information and legal rules. All this data need to be analyzed in a coordinated manner to produce insights that will save cost both for the health care provider and care receiver while remaining legally compliant.

1.2.4 Computer Vision:

The advancement in recognizing an image by a computer involves processing large sets of image data from multiple objects of same category. For example, face recognition. These data sets are modeled, and algorithms are created to apply the model to newer images to get a satisfactory result. Processing of these huge data sets and creation of models need various tools used in Data science.

1.2.5 Efficient Management of Energy:

As the demand for energy consumption soars, the energy producing companies need to manage the various phases of the energy production and distribution more efficiently. This involves optimizing the production methods, the storage and distribution mechanisms as well as studying the customers consumption patterns. Linking the data from all these sources and deriving insight seems a daunting task. This is made easier by using the tools of data science.

1.2.6 Java in Data Science:

The programming requirement of data science demands a very versatile yet flexible language which is simple to write the code but can handle highly complex mathematical processing. Java is most suited for such requirements as it has already established itself both as a language for general computing as well as scientific computing. Moreover it is being continuously upgraded in form of new addition to its plethora of libraries aimed at different programming requirements. Below we will discuss such features of java which makes it the preferred language for data science.

A simple and easy to learn language which achieves result in fewer lines of code than other similar languages like java. Its simplicity also makes it robust to handle complex scenarios with minimal code and much less confusion on the general flow of the program.

It is cross platform, so the same code works in multiple environments without needing any change. That makes it perfect to be used in a multi-environment setup easily.

It executes faster than other similar languages used for data analysis like R and Its excellent memory management capability, especially garbage collection makes it versatile in gracefully managing very large volume of data transformation, slicing, dicing and visualization.

Most importantly Java has got a very large collection of libraries which serve as special purpose analysis tools. For example – the Numbly package deals with scientific computing and its array needs much less memory than the conventional

Java list for managing numeric data. And the number of such packages is continuously growing.

Java has packages which can directly use the code from other languages like Java or C. This helps in optimizing the code performance by using existing code of other languages, whenever it gives a better result.

In the subsequent chapters we will see how we can leverage these features of java to accomplish all the tasks needed in the different areas of Data Science.

1.3 SYSTEM ARCHITECTURE:

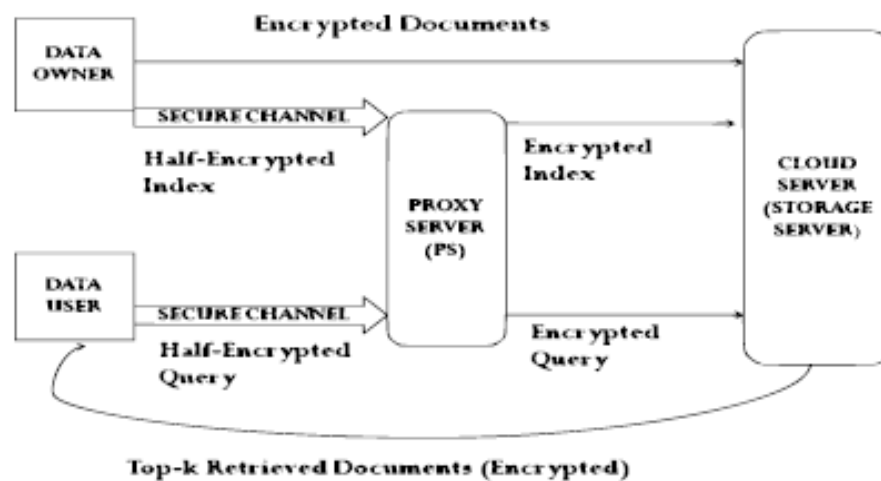


Fig 1.1 System Architecture

2. LITERATURE SURVEY

21 A kNN QUERY PROCESSING ALGORITHM USING A TREE INDEX STRUCTURE ON THE ENCRYPTED DATABASE

AUTHORS:Hyeong-Il Kim, Hyeong-Jin Kim, Jae-Woo Chang

With the adoption of cloud computing, database outsourcing has emerged as a new platform. Due to the serious privacy concerns in the cloud, database need to be encrypted before being outsourced to the cloud. Therefore, various kNN query processing techniques have been proposed over the encrypted database. However, the existing schemes are either insecure or inefficient. So, we, in this paper, propose a new secure kNN query processing algorithm. Our algorithm guarantees the confidentiality of both the encrypted data and a user's query record. To achieve the high query processing efficiency, we also devise an encrypted index search scheme which can perform data filtering without revealing data access patterns. We show from our performance analysis that the proposed scheme outperforms the existing scheme in terms of a query processing cost while preserving data privacy.

22 EFFICIENT QUERY PROCESSING ON OUTSOURCED ENCRYPTED DATA IN CLOUD WITH PRIVACY PRESERVATION

AUTHORS: B.R. Purushothama, B.B. Amberker

Data outsourcing on to the public cloud faces several security challenges. Ensuring the confidentiality of the outsourced sensitive data is of paramount importance for the adoption of public cloud for data outsourcing. Often the cloud storage servers are entrusted. Encryption method is used for maintaining the confidentiality of the outsourced data. Performing the queries on the encrypted data is a challenging task. Adversary should not gain any significant information other than the minimal information by observing the queries and the query responses. In this work, we provide two solutions which are efficient in

processing the queries on the encrypted data. We focus on improving the performance of the query processing without compromising the privacy of the data and the queries. We show that adversary can gain no significant information about the data other than the minimal information which cannot be avoided. We conduct the empirical performance evaluations and compare with the scheme available in the literature. Our experiments show that the proposed schemes are efficient in comparison with the existing scheme.

23 HILBERT-CURVE BASED CRYPTOGRAPHIC TRANSFORMATION SCHEME FOR PROTECTING DATA PRIVACY ON OUTSOURCED PRIVATE SPATIAL DATA

AUTHORS: Hyeong-Il Kim, Seung-Tae Hong, Jae-Woo Chang

The study on spatial database outsourcing has been spotlighted with the development of cloud computing. Therefore, researches for protecting location data privacy in outsourced database have been studied. However, the existing spatial transformation schemes are vulnerable to naïve attack models. The existing cryptographic transformation scheme provides high data privacy, but it causes the high query processing cost. To solve these problems, in this paper we propose a Hilbert-curve based cryptographic transformation scheme to protect data privacy and to improve the efficiency of the query processing in outsourced databases. The proposed scheme reduces the communication cost for query processing since we perform a local clustering based on the order of Hilbert-curve. Finally, we show from performance analysis that the proposed scheme outperforms the existing scheme.

24 CONTROLLING OUTSOURCING DATA IN CLOUD COMPUTING WITH ATTRIBUTE-BASED ENCRYPTION

AUTHORS: Shuaishuai Zhu, Yiliang Han, Yuechuan Wei

In our IT society, cloud computing is clearly becoming one of the dominating infrastructures for enterprises as long as end users. As more cloud based services available to end users, their oceans of data are outsourced in the cloud as well.

Without any special mechanisms, the data may be leaked to a third party for unauthorized use. Most presented works of cloud computing put these emphases on computing utility or new types of applications. But in the view of cloud users, such as traditional big companies, data in cloud computing systems is tend to be out of control and privacy fragile. So most of data they outsourced is less important. A mechanism to guarantee the ownership of data is required. In this paper, we analyzed a couple of recently presented scalable data management models to describe the storage patterns of data in cloud computing systems. Then we defined a new tree-based dataset management model to solve the storage and sharing problems in cloud computing. A couple of operation strategies including data encryption, data boundary maintenance, and data proof are extracted from the view of different entities in the cloud. The behaviors of different users are controlled by view management on the tree. Based on these strategies, a flexible data management mechanism is designed in the model to guarantee entity privacy, data availability and secure data sharing.

25 AUTHORIZED PRIVATE KEYWORD SEARCH OVER ENCRYPTED DATA IN CLOUD COMPUTING

AUTHORS: Ming Li, Shucheng Yu, Ning Cao

In cloud computing, clients usually outsource their data to the cloud storage servers to reduce the management costs. While those data may contain sensitive personal information, the cloud servers cannot be fully trusted in protecting them. Encryption is a promising way to protect the confidentiality of the outsourced data, but it also introduces much difficulty to performing effective searches over encrypted information. Most existing works do not support efficient searches with complex query conditions, and care needs to be taken when using them because of the potential privacy leakages about the data owners to the data users or the cloud server. In this paper, using on line Personal Health Record (PHR) as a case study, we first show the necessity of search capability authorization that reduces the privacy exposure resulting from the search results,

and establish a scalable framework for Authorized Private Keyword Search (APKS) over encrypted cloud data. We then propose two novel solutions for APKS based on a recent cryptographic primitive, Hierarchical Predicate Encryption (HPE). Our solutions enable efficient multi-dimensional keyword searches with range query; allow delegation and revocation of search capabilities. Moreover, we enhance the query privacy which hides users' query keywords against the server. We implement our scheme on a modern workstation, and experimental results demonstrate its suitability for practical usage.

26 EXISTING SYSTEM:

The first Searchable Encryption SE scheme was proposed by D. X. Song using symmetric key encryption algorithm. SE by public key based approach was proposed by using Identity-Based Encryption (IBE). BE scheme allows multiple users to perform the search operation over the encrypted data. Another scheme supporting multiple users' search operation is proposed by using CP-ABE. Keyword authorization based approach supports search operation by multiple users. Multi-Keyword Ranked Search approach over the data of multiple owners is proposed. This approach supports search operation in a multi-owner and multi-user environment, which allows multiple users to perform the search operation over the data of multiple owners.

27 DISADVANTAGES OF EXISTING SYSTEM:

- These approaches support search operation in a single owner and a single user environment, which allows only a single user to perform the search operation over the data of single owner.
- All these schemes support search operation in a single owner and multiuser environment, which allows the multiple users to perform the search operation over the encrypted data of a single owner.

3. PROPOSED SYSTEM

Cloud server is assigned the task of storing all the documents and indices from different owners and when a search request from a data user is received, it needs to find the most relevant documents and return them to the data user. A data owner creates an index for each of its documents. It encrypts the document collection and sends the encrypted documents over to the cloud server. The words in the indices are partially encrypted with the owners secret key and then these indices are sent to the proxy server. A proxy server is given the work of completing the encryption of partially encrypted index words as well as query keywords before they are sent to the cloud server. The proxy server has a key, known to only it, that is used as a common key to complete the encryption of all the partially encrypted words received. A data user's task is to frame search queries and to partially encrypt these query keywords with its own secret key before sending them to the proxy server.

3.1 ADVANTAGES OF PROPOSED SYSTEM:

- Query Transformation Elimination: To allow the multiple users to perform the search operation over the data of multiple owners without transforming the queries.
- Top-k Retrieval: To return the top-k relevant documents to the users' queries by using Euclidean distance similarity approach. Sending top-k documents helps the data users in fulfilling their requirements quickly by going through the top-k documents only and it also avoids causing unnecessary network traffic.
- Privacy of Information: To prevent the information leakages from the encrypted indices and trapdoors and also to prevent the direct possible inferences, i.e., guessing keywords of the indices from the relevance score information present in them.

3.2 TECHNIQUES:

There Are Two Techniques:

1. Term Frequency
2. Inverse Document Frequency

3.2.1 Term Frequency:

Term frequency (TF) often used in Text Mining, NLP and Information Retrieval tells you how frequently a term occurs in a document. In the context natural language, terms correspond to words or phrases. Since every document is different in length, it is possible that a term would appear more often in longer documents than shorter ones. Thus, term frequency is often divided by 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).

There is a way you can improve the ranking of these words such that topic words appear more prominently. The first approach is to eliminate all stop words (common words) such as 'the', 'is', 'are' and so on before computing the term frequencies. Some of the stop words removed where the larger fonts indicate high term frequencies. Term frequency (TF) is only one part of the TF-IDF approach to information retrieval.

- Term frequency often used in text mining and information retrieval tells you how frequency a term occurs in a document.
- Since every document is different length it is possible that a term would appear more often in longer documents than shorter once.
- Removed the larger fonts indicate high term frequency.

3.2.2 Inverse Document Frequency:

The suppress common words and surface topic words is to multiply the term frequencies with what's called Inverse Document Frequencies (IDF). IDF is a weight indicating how widely a word is used. The more frequent its usage across documents, the lower its score. For example, the word they would appear in almost all English texts and thus would have a very low inverse document frequency.

IDF = (Total number of documents / Number of documents with word t in it)

Multiplying term frequencies with the IDFs dampens the frequencies of highly occurring words and improves the prominence of important topic words and this is the basis of the commonly talked about weighting Document frequency measures commonness and we prefer to measure rareness. The classic way that this is done is with a formula

$$idf_j = \log \left[\frac{n}{df_j} \right]$$

- Inverse document frequency is a statistical weight used for measuring the importance of a term in a text document collection.
- The document frequency is defined by the number of documents in which a term appears.
- The terms with low document frequency more valuable than terms with high document frequency.

TF-IDF is the product of TF and IDF:

TF-IDF = TF * IDF

In order to acquire good results with TF-IDF, a huge corpus is necessary. I just used a small sized corpus. Since I removed stop words, result was pleasant.

3.3 Functional and Non Functional Requirement:

Functional Requirements:

- The Data Owners Register to the proxy server and upload to the owners document with half encrypted send to the proxy server.
 - The Data User Register to the Cloud server and search a keyword to the cloud server get to the file.
 - The Proxy server receives the dataowners file and that file is fully encrypted send to the cloud.
 - Cloud server is assigned the task of storing all the documents and indices from different owners and when a search request from a data user is received.
-

Non-Functional Requirements:

The system should be support Net beans. Each member should have a separate system. The system should have Role based System functions access. Approval Process has to be defined. The system should have Modular customization components so that they can be reused across the implementation. These are the mainly following:

- Secure access of confidential data (expense details). 24 X 7 availability
- Better component design to get better performance at peak time
- Flexible service based architecture will be highly desirable for future extension

3.4 SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

- System : Pentium Dual Core.
- Hard Disk : 500 GB.
- Monitor : 15” LED
- Input Devices : Keyboard, Mouse
- Ram : 1GB.

SOFTWARE REQUIREMENTS:

- Operating system : Windows 7.
- Coding Language : JAVA/J2EE
- Tool : Netbeans 7.2.1
- Database : MYSQL

4. SYSTEM DIAGRAMS

4.1 SYSTEM ARCHITECTURE:

Searchable Encryption (SE) schemes provide security and privacy to the cloud data. This scheme is allowing multi users and multi owners upload files and download multi users.

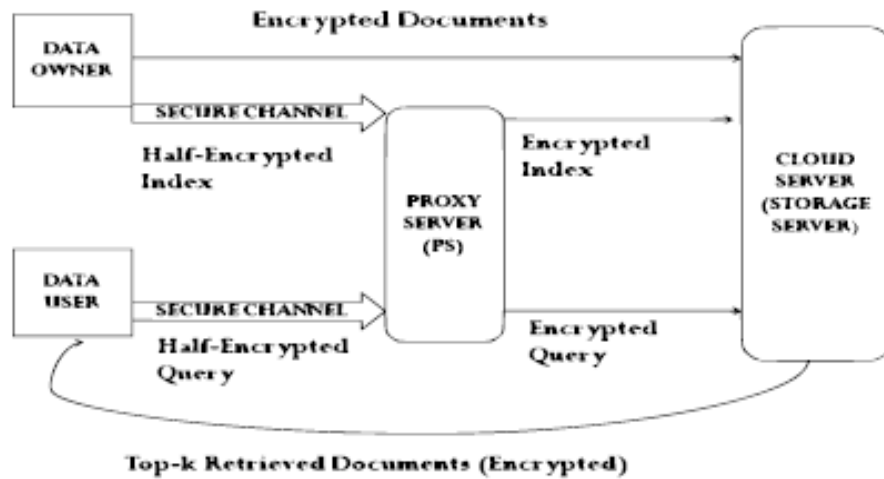


Fig 4.1 System Architecture

4.2 MODULE DESCRIPTION:

Number of Modules

After careful analysis the system has been identified to have the following modules:

1. Building Index
2. Index Encryption
3. Search Operation
4. Matching Score Calculation

MODULES DESCRIPTION:

Building Index

The data owners create an index for each of their documents as follows:
Initially the stop words and non-alphabetic characters in documents are identified and

ignored. Unique keywords in each document are listed and corresponding TF-IDF values are noted. The TF-IDF is a keyword scoring mechanism, which conveys the importance of the keyword in the entire data set. Hence, they are referred to as the relevance score information. The TF-IDF includes two attributes called Term Frequency and Inverse Document Frequency.

TF refers to the number of occurrences of term 't' in a document 'd'. The IDF is obtained by dividing the total number of documents by the Document Frequency (DF). The DF is the number of documents that contain the term 't'. To avoid any direct possible inferences from the TFIDF values by the cloud server, each TF-IDF value is raised to the power of a random number. For example, suppose TF-IDF values of two keywords are 3, 2 and the random number is chosen as 3. The new TF-IDF values will become 27 and 9. The information as to which users are permitted to search an index is also included in the index by maintaining a list of all the user ids permitted to access the index.

Index Encryption

At the Data owner side For each keyword of the index, determine its length (n). If the length is even, randomly select $n/2$ number of positions within the keyword and encrypt the characters located at those positions using RSA algorithm with the private key of the data owner. For odd length keyword, encrypt $n/2+1$ character randomly using the private key of owner. Once the above is done for every keyword in an index, then the partially encrypted index is sent to the proxy server. At the Proxy server side, the proxy server receives the partially encrypted index. It has its own secret key called as the common key, which is used to encrypt the remaining unencrypted characters of each keyword in the index. Thus it completes the index encryption fully.

Search Operation

To retrieve the relevant documents, the data user issues his/her user id and a

query, which is required to be encrypted. The data user randomly selects the positions of the characters within each keyword for encryption. The encryption of each keyword of the query follows the same procedure as explained in index encryption. The queries after encryption are termed as trapdoors. The trapdoors and the corresponding user ids along with the parameter 'k' are sent to the cloud server. The cloud server then checks if the user is authorized to search an index. If the user is authorized, trapdoors are matched with the keywords of the index and a match score is calculated, which is explained below.

Matching Score Calculation

Every keyword in query is matched with each keyword in each index. This matching is done by making use of the Euclidean distance similarity approach. The least match score implies the highest match. The matching score is calculated as follows , Initially the lengths of keywords (index keyword and query keyword) that are to be matched are found. If the lengths are found to be different, current index keyword is ignored. When the lengths are same, sum of squares of differences in ASCII values of letters in the first keyword and corresponding letters in the second keyword is found. The square root of this value is called the match score.

4.3 UML DIAGRAMS

- UML stands for Unified Modeling Language. UML is a standardized general- purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.
- The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

- The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.
- The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.
- The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

4.3.1 USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Use Case Diagram

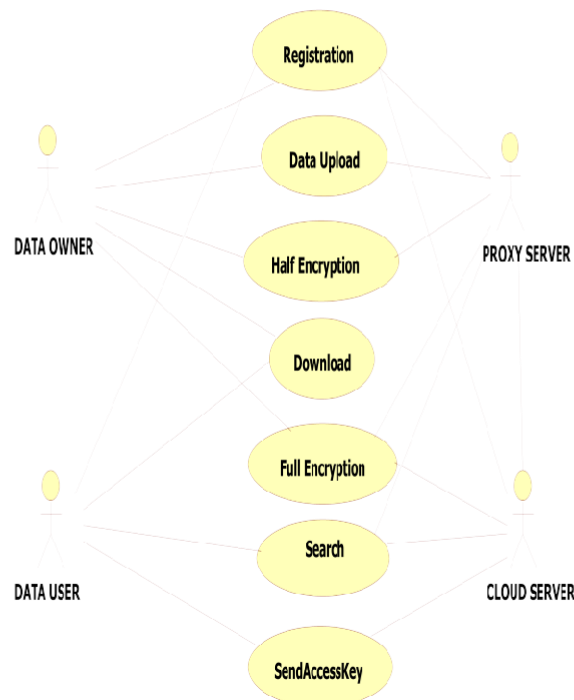


Fig 4.3.1 Use Case Diagram

4.3.2 CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

Class Diagram:

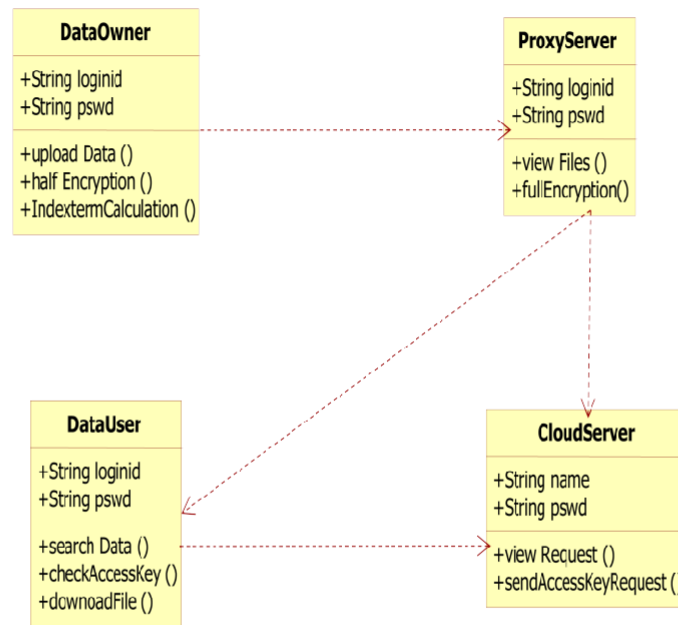


Fig 4.3.2 Class Diagram

4.3.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

Sequence diagram:

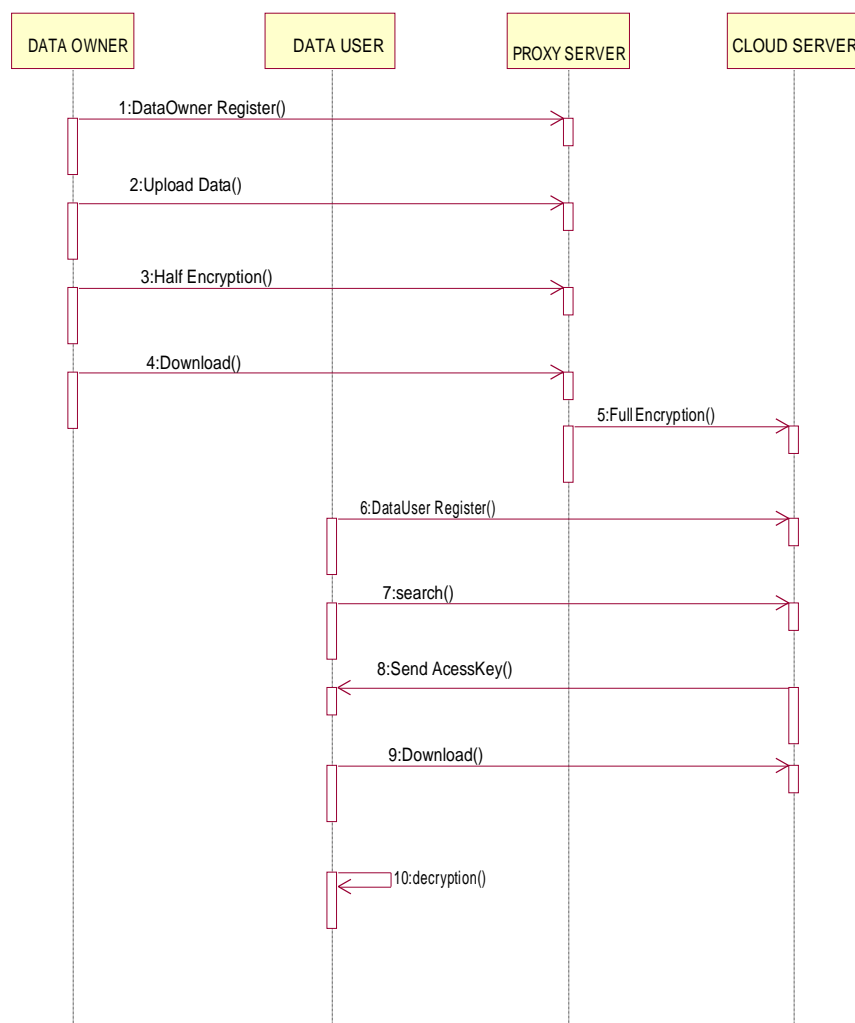


Fig 4.3.3 Sequence Diagram

4.3.4 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

Activity Diagram:

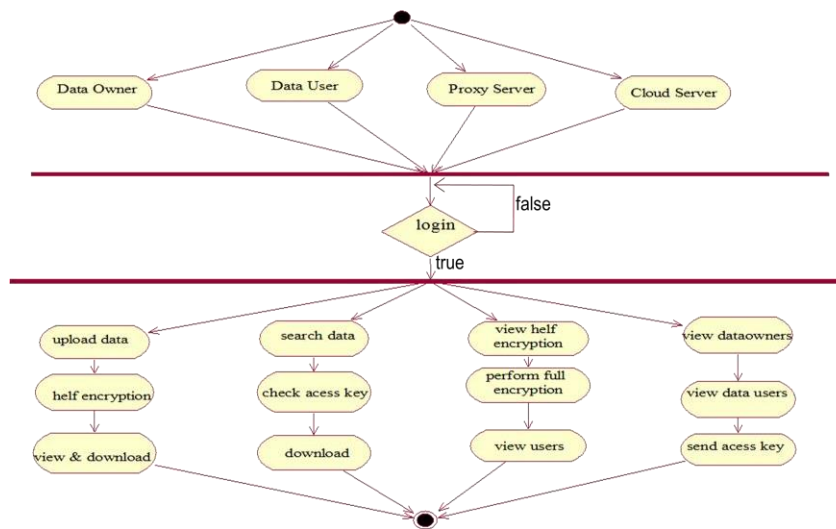


Fig 4.3.4 Activity Diagram

4.3.5 COLLABORATION DIAGRAM:

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object.

Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements. Several vendors offer software for creating and editing collaboration diagrams

Collaboration Diagram:

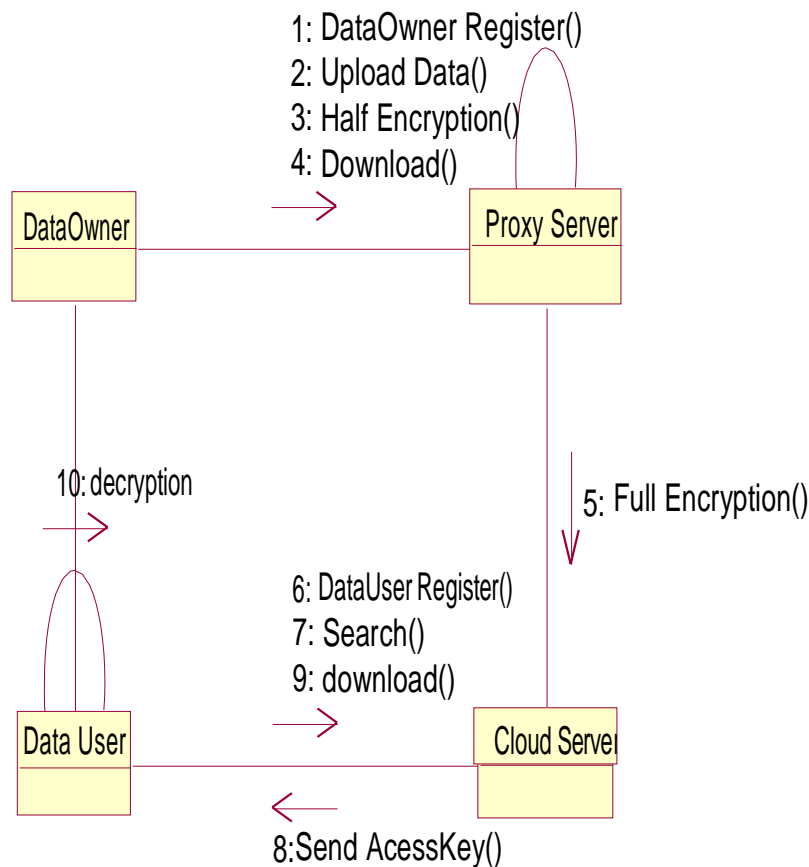


Fig 4.3.5 Collaboration Diagram

4.3.6 COMPONENT DIAGRAM:

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system.

Component Diagram:

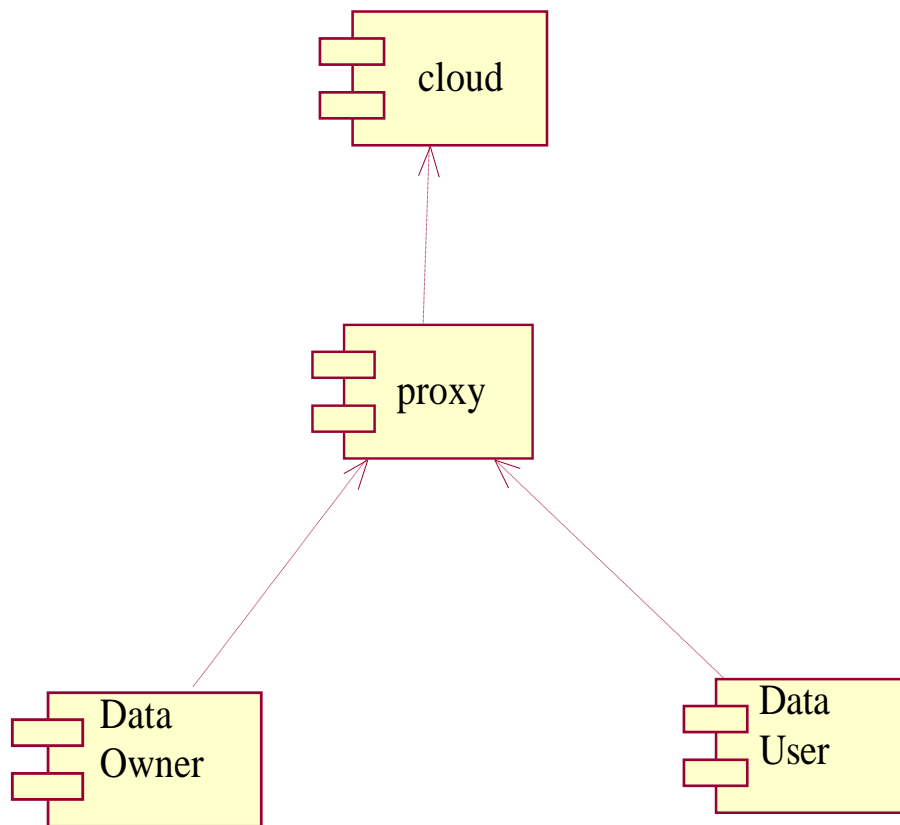


Fig 4.3.6 Component Diagram

4.3.7 DEPLOYMENT DIAGRAM:

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of system. Deployment diagrams consist of nodes and their relationships. The term Deployment itself describes the purpose of the diagram. Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system. Deployment diagrams are used by the system engineers. The purpose of deployment diagrams can be described as – Visualize the hardware topology of a system.

Deployment Diagram:

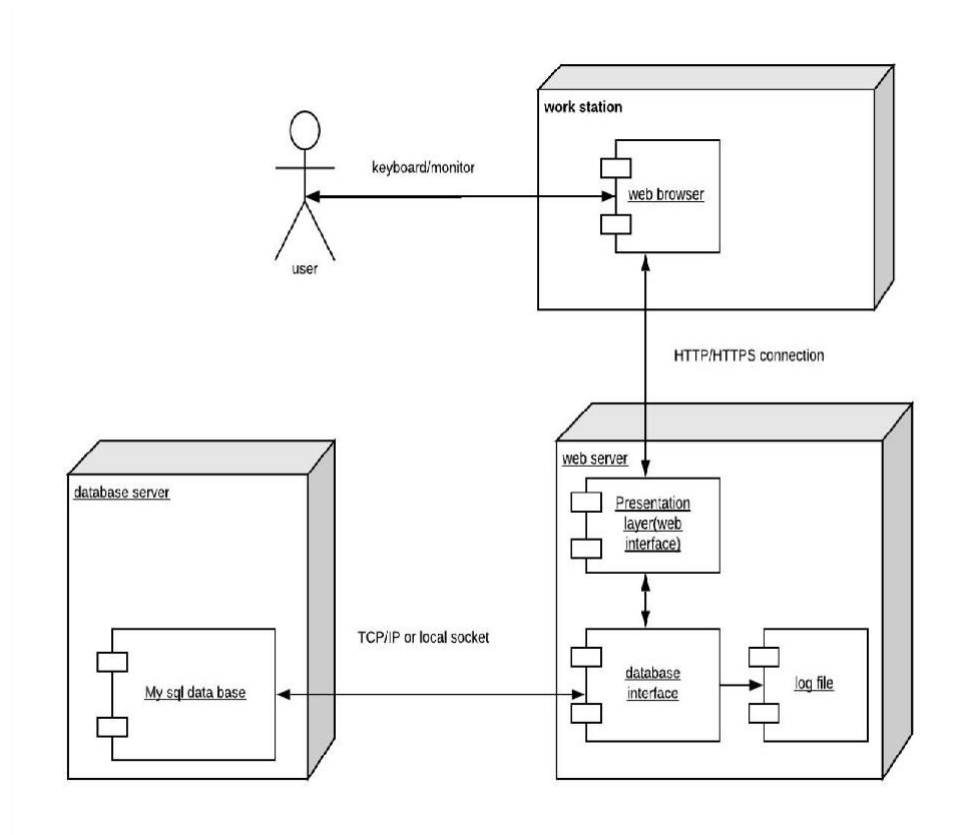


Fig 4.3.7 Deployment Diagram

4.4 Data Dictionaries:

In SQL Server the data dictionary is a set of database tables used to store information about a database's definition. The dictionary contains information about database objects such as tables, indexes, columns, **data types**, and views.

The data dictionary is used by SQL Server to execute queries and is automatically updated whenever objects are added, removed, or changed within the database.

DATA OWNER:

| Table | Create Table |
|---------------|---|
| data owner | <pre>CREATE TABLE `dataowner` (`id` int(11) NOT NULL auto_increment,`loginname` varchar(50) NOT NULL, `pswd` varchar(50) NOT NULL, `email` varchar(50) NOT NULL, `mobile` varchar(50) NOT NULL, `city` varchar(50) NOT NULL,`state` varchar(50) NOT NULL, PRIMARY KEY (`id`),UNIQUE KEY `loginname` (`loginname`) UNIQUE KEY `email` (`email`), UNIQUE KEY `loginname_2` (`loginname`,`email`)) ENGINE=InnoDB DEFAULT CHARSET=latin1</pre> |

**Table 4.1 DDL Information For
qerytransformation.dataowner**

| Table | No n_u niq ue | Key nam e | Seq_ in_in dex | Colu mn_n ame | Collat ion | Card inalit y | Sub_ part | Pack ed | N ul l | Inde x_ty pe |
|---------------|------------------------|-------------------------|----------------------|---------------------|---------------|---------------------|--------------|------------|--------------|--------------------|
| Data owner | 0 | PRI MA RY | 1 | Id | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data owner | 0 | Logi n name | 1 | Login name | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data owner | 0 | email | 1 | email | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data owner | 0 | Logi n name | 1 | email | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data owner | 0 | Logi n name _2 | 1 | loginn ame_2 | A | 1 | (NU LL) | (NU LL) | | BTR EE |

**Table4.2 Index Information for
- qerytransformation.dataowner**

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|-----------|-------------|-------------------|------|-----|---------|----------------|---------------------------------|
| id | int(11) | (NULL) | NO | PRI | (NULL) | auto_increment | select,insert,update,references |
| Loginname | varchar(50) | latin1_swedish_ci | NO | UNI | | | select,insert,update,references |
| Pwsd | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |
| Email | varchar(50) | latin1_swedish_ci | NO | UNI | | | select,insert,update,references |
| Mobile | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |
| City | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |

**Table4.3 Column Information For
qerytransformation.dataowner**

DATA USERS:

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|-----------|-------------|-------------------|------|-----|---------|----------------|---------------------------------|
| id | int(11) | (NULL) | NO | PRI | (NULL) | auto_increment | select,insert,update,references |
| Loginname | varchar(50) | latin1_swedish_ci | NO | UNI | | | select,insert,update,references |
| Pwsd | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |
| Email | varchar(50) | latin1_swedish_ci | NO | UNI | | | select,insert,update,references |
| Mobile | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |
| City | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |

**Table4.4 Column Information For
qerytransformation.datauser**

| Table | No n_u niq ue | Key nam e | Seq_ in_in dex | Colu mn_n ame | Collat ion | Card inalit y | Sub_ part | Pack ed | N ul l | Inde x_ty pe |
|--------------|------------------------|-------------------------|----------------------|---------------------|---------------|---------------------|--------------|------------|--------------|--------------------|
| Data User | 0 | PRI MA RY | 1 | Id | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data User | 0 | Logi n name | 1 | Login name | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data User | 0 | email | 1 | email | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data User | 0 | Logi n name | 1 | email | A | 1 | (NU LL) | (NU LL) | | BTR EE |
| Data User | 0 | Logi n name _2 | 1 | loginn ame_2 | A | 1 | (NU LL) | (NU LL) | | BTR EE |

**Table4.5 Index Information for
- qerytransformation.datauser**

| Table | Create Table |
|--------------|---|
| data user | <pre>CREATE TABLE `datauser` (`id` int(11) NOT NULL auto_increment,`loginname` varchar(50) NOT NULL, `pwsd` varchar(50) NOT NULL, `email` varchar(50) NOT NULL, `mobile` varchar(50) NOT NULL, `city` varchar(50) NOT NULL,`state` varchar(50) NOT NULL, PRIMARY KEY (`id`),UNIQUE KEY `loginname` (`loginname`) UNIQUE KEY `email` (`email`), UNIQUE KEY `loginname_2` (`loginname`,`email`)) ENGINE=InnoDB DEFAULT CHARSET=latin1</pre> |

**Table 4.6 DDL Information For
qerytransformation.datauser**

OWNERS FILE:

| Field | Type | Collation | Null | Key | Default | Extra | Privileges |
|-----------|-------------|-------------------|------|-----|---------|----------------|---------------------------------|
| id | int(11) | (NULL) | NO | PRI | (NULL) | auto_increment | select,insert,update,references |
| Loginname | varchar(50) | latin1_swedish_ci | NO | UNI | | | select,insert,update,references |
| Pwsd | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |
| Email | varchar(50) | latin1_swedish_ci | NO | UNI | | | select,insert,update,references |
| Mobile | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |
| City | varchar(50) | latin1_swedish_ci | NO | | | | select,insert,update,references |

**Table4.7 Column Information For
querytransformation.ownersFile**

| Tab le | Non _uni que | Key na me | Se q_i n_i nd ex | Colu mn_n ame | Collat ion | Cardin ality | Sub_p art | Pack ed | Nu ll | Inde x_ty pe |
|-------------------|-----------------------------|--------------------------|---|------------------------------|-----------------------|-------------------------|----------------------|--------------------|------------------|-----------------------------|
| Data own er | 0 | PRI MA RY | 1 | Id | A | 1 | (NUL L) | (NU LL) | | BTR EE |

**Table4.8 Index Information For
- Qerytransformation.Owners File**

5. IMPLEMENTATION

IMPLEMENTATION:

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.1 SIMPLE CODE:

5.1.1 DBConnection.java

```
package com.transformation.db;
import java.sql.Connection;
import java.sql.DriverManager;
/**
 *
 * @author Ramu Maloth
 */
public class DBConnection {
    public static Connection con = null;
    public static Connection getDBConnection(){
        try {
            DriverManager.registerDriver(new com.mysql.jdbc.Driver());
```

```

Con=DriverManager.getConnection("jdbc:mysql://localhost:3306/qerytransform
ation","root","root");
if(con!=null){
return con;
}
} catch (Exception e) {
System.out.println("Error at DBConnection "+e.getMessage());
}
return con;
}

```

5.1.2 Ownerupload.Java

```

package com.transformation.actions;
import com.transformation.db.DBConnection;
import com.transformation.util.EncryptionAlgoritham;
import com.transformation.util.MainDocumentLogic;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

```

```

import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import javax.servlet.http.Part;
/**
 *
 * @author Ramu Maloth
 */
@MultipartConfig
public class OwnerUpload extends HttpServlet {
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
HttpSession hs = request.getSession();
String ownername = hs.getAttribute("oname").toString();
String oemail = hs.getAttribute("email").toString();
String privateKey = request.getParameter("privatekey");
Part filePart = request.getPart("file");
InputStream inputStream = null;
BufferedReader reader = null;
    StringBuffer buf;
int repeatedFrequency = 0;
    int termFrequency = 0;
    StringBuffer streamingIndexes = new StringBuffer();
    StringBuffer termFrequencyData = new StringBuffer();
Connection con = null;
    PreparedStatement ps = null;
String fileName = getFileName(filePart);
if (filePart != null) {
    inputStream = filePart.getInputStream();
}

```

```

String str = "";
buf = new StringBuffer();
reader = new BufferedReader(new InputStreamReader(inputStream));
if (inputStream != null) {
    while ((str = reader.readLine()) != null) {
        buf.append(str + "\n");
    }
}
String data = buf.toString();
Map map = MainDocumentLogic.getDocumentIndexes(data);
//System.out.println(map);
Set set = map.entrySet();
Iterator it = set.iterator();
while(it.hasNext()){
    Map.Entry me = (Map.Entry)it.next();
    String value = (String)me.getValue();
    int valNo = Integer.parseInt(value);
    if(valNo == 1){
        String words = (String)me.getKey();
        String keys = (String)me.getValue();
        streamingIndexes.append(words+" ");
        termFrequencyData.append(words+" : "+keys+"\n");
        termFrequency+= Integer.parseInt(keys);
    }else{
        String keys = (String)me.getValue();
        repeatedFrequency+= Integer.parseInt(keys);
    }
}

final int mid = data.length() / 2; //get the middle of the String
String[] parts = {data.substring(0, mid),data.substring(mid)};

```

```

String firstHalf = parts[0];
String secondHalf = parts[1]; //second part
String stmIndex = streamingIndexes.toString();
String firsthalfEnc = null;
try {
    firsthalfEnc = EncryptionAlgorithm.encrypt(stmIndex);
} catch (Exception e) {
    System.out.println("Encryption Error "+e);
}
String streamIndexWithHalfEncryption = firsthalfEnc + " "+secondHalf;
String trmFrequency = termFrequencyData.toString();
float trmFrnc = (float)termFrequency/repeatedFrequency;
try {
    con = DBConnection.getDBConnection();
    String sqlQuery = "insert into
ownersfiles(ownername,oemail,filename,originaldata,streamingindexes,termfreq
uency,halfencryption,privatekey,frequency) values(?,?,?,?,?,?,?,?)";
    ps = con.prepareStatement(sqlQuery);
    ps.setString(1, ownername);
    ps.setString(2, oemail);
    ps.setString(3, fileName);
    ps.setString(4, data);
    ps.setString(5, stmIndex);
    ps.setString(6, trmFrequency);
    ps.setString(7, streamIndexWithHalfEncryption);
    ps.setString(8, privateKey);
    ps.setFloat(9, trmFrnc);
    int no = ps.executeUpdate();
    if(no > 0){
response.sendRedirect("OwnerFileUploadSuccess.jsp?stmIndex="+stmIndex+"

```

```

&trmFrequency="+trmFrequency+"&streamIndexWithHalfEncryption="+streamIndexWithHalfEncryption);
    }else{
        response.sendRedirect("OwnerFileUploadSuccess.jsp?msg=faild");
    }
} catch (Exception e) {
    e.printStackTrace();
    response.sendRedirect("OwnerFileUploadSuccess.jsp?msg=faild");
}
}

private String getFileName(Part filePart) {
    for (String cd : filePart.getHeader("content-disposition").split(";")) {
        if (cd.trim().startsWith("filename")) {
            return cd.substring(cd.indexOf('=') + 1).trim().replace("\"", "");
        }
    }
    return null;
}
}

```

5.1.3 Encryptionalgorithm.Java

```

package com.transformation.util;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;
public class EncryptionAlgorithm {
    private static final String ALGO = "AES";
    private static final byte[] keyValue =

```

```

        new byte[] { 'T', 'h', 'e', 'B', 'e', 's', 't', 'S', 'e', 'c', 'r', 'e', 't', 'K', 'e', 'y' };
    public static String encrypt(String Data) throws Exception {
        Key key = generateKey();
        Cipher c = Cipher.getInstance(ALGO);
        c.init(Cipher.ENCRYPT_MODE, key);
        byte[] encVal = c.doFinal(Data.getBytes());
        String encryptedValue = new BASE64Encoder().encode(encVal);
        return encryptedValue;}

    public static String decrypt(String encryptedData) throws Exception {
        Key key = generateKey();
        Cipher c = Cipher.getInstance(ALGO);
        c.init(Cipher.DECRYPT_MODE, key);
        byte[] decordedValue=new
BASE64Decoder().decodeBuffer(encryptedData);
        byte[] decValue = c.doFinal(decordedValue);
        String decryptedValue = new String(decValue);
        return decryptedValue; }

    private static Key generateKey() throws Exception {
        Key key = new SecretKeySpec(keyValue, ALGO);
        return key;
    }
}

```

5.1.4 Acckeycheck.Java

```

package com.transformation.util;
import com.transformation.db.DBConnection;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
/**

```



```

*
* @author Ramu Maloth
*/

public class AccKeyCheck {

    public boolean checkKey(int fileid,String filename,String accekey){
        boolean flag = false;

        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        int count = 0;
        try {
            con = DBConnection.getDBConnection();
            String sqlQuery = "select count(*) from userrequest where fileid = ? and
filename = ? and accesskey = ?";
            ps = con.prepareStatement(sqlQuery);
            ps.setInt(1, fileid);
            ps.setString(2, filename);
            ps.setString(3, accekey);
            rs = ps.executeQuery();
            if(rs.next()){
                count = rs.getInt(1);
            }if(count > 0){
                flag = true;
            }else{
                flag = false;
            }
        } catch (Exception e) {
            System.out.println("Access Key Check Error "+e.getMessage());
        }finally{
            try {

```

```

        rs.close();
        ps.close();
        con.close();
    } catch (Exception e) {
    } }
return flag; }

public String checkGeneratedKey(int fileid,String email){
    boolean flag = false;
    Connection con = null;
    PreparedStatement ps = null;
    ResultSet rs = null;
    String accessKey = null;
    try {
        con = DBConnection.getDBConnection();
        String sqlQuery = "select accesskey from userrequest where fileid = ?
and usermail = ?";

        ps = con.prepareStatement(sqlQuery);
        ps.setInt(1, fileid);
        ps.setString(2, email);
        rs = ps.executeQuery();
        if(rs.next()){
            accessKey = rs.getString(1);
        }
    } catch (Exception e) {
        System.out.println("Access Key Check Error "+e.getMessage());
    }finally{
        try {
            rs.close();
            ps.close();
            con.close();

```

```

        } catch (Exception e) {
        } }
return accessKey;}

public void updateAccessKey(int fileid,String email,String accesskey){
boolean flag = false;
Connection con = null;
PreparedStatement ps = null;
ResultSet rs = null;
try {
    con = DBConnection.getDBConnection();
    String sqlQuery = "update userrequest set accesskey = ?,status = ? where
fileid = ? and usermail = ?";
    ps = con.prepareStatement(sqlQuery);
    ps.setString(1, accesskey);
    ps.setString(2, "accepted");
    ps.setInt(3, fileid);
    ps.setString(4, email);
    ps.executeUpdate();
} catch (Exception e) {
    System.out.println("Access Key Update Error "+e.getMessage());
}finally{
try {
    rs.close();
    ps.close();
    con.close();
} catch (Exception e) {
}
}
}

```

6. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

6.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.3 Functional Test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box

tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.7 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results:

All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

6.8 Test case:

| S.no | Test Case | Excepted Result | Result | Remarks(IF Fails) |
|------|-------------------------|--|--------|--|
| 1 | Data owner Registration | If Data owner registration successfully | Pass | If Data Owner is not registered. |
| 2 | User Registration | If User registration successfully. | Pass | If User is not registered. |
| 3 | Data owner Login | If Data owner name and password is correct then it will getting valid page | Pass | If Data owner name or password is not correct. |
| 4 | Data user Login | If user name and password is correct then it will getting valid page. | Pass | If user name or password is not correct. |
| 5 | Proxy Login | If Proxy server name and | Pass | If user name or password is |

| | | | | |
|----|----------------|--|------|---|
| | | password is correct then it will getting valid page. | | not correct. |
| 6 | Cloud Login | If admin name and password is correct then it will getting valid page. | Pass | If admin name or password is not correct. |
| 7 | Owner Homepage | View all owner files | Pass | If Data owner is not valid |
| 8 | File upload | Upload files here | Pass | If uploaded files is not valid |
| 9 | Files view | View all files | Pass | If view files are not valid |
| 10 | User Homepage | View all User Files | Pass | If user is not valid. |
| 11 | File search | File is found | Pass | If file is not valid |

| | | | | |
|----|-------------------|---|------|--|
| 12 | File request | File is found | Pass | If file is not valid |
| 13 | File download | File is downloaded | Pass | If file is not valid |
| 14 | Proxy Homepage | View all data users and data owners details | Pass | If data users and data owners is valid |
| 15 | View data owner | View data owner details | Pass | If data owner is valid |
| 16 | View data user | View data user details | Pass | If data user is valid |
| 17 | View owners files | View data owners files | pass | If owners file is not valid |
| 18 | Cloud Homepage | View all cloud details | Pass | If cloud is not valid |
| 19 | View data owner | View data owner details | Pass | If data owner is valid |

| | | | | |
|----|------------------|------------------------|------|-----------------------|
| 20 | View data user | View data user details | Pass | If data user is valid |
| 21 | File information | View file information | Pass | If file is valid |
| 22 | Sending keys | Keys will be generated | Pass | If Email is not valid |

7. SCREEN SHOTS

7.1 Home Page

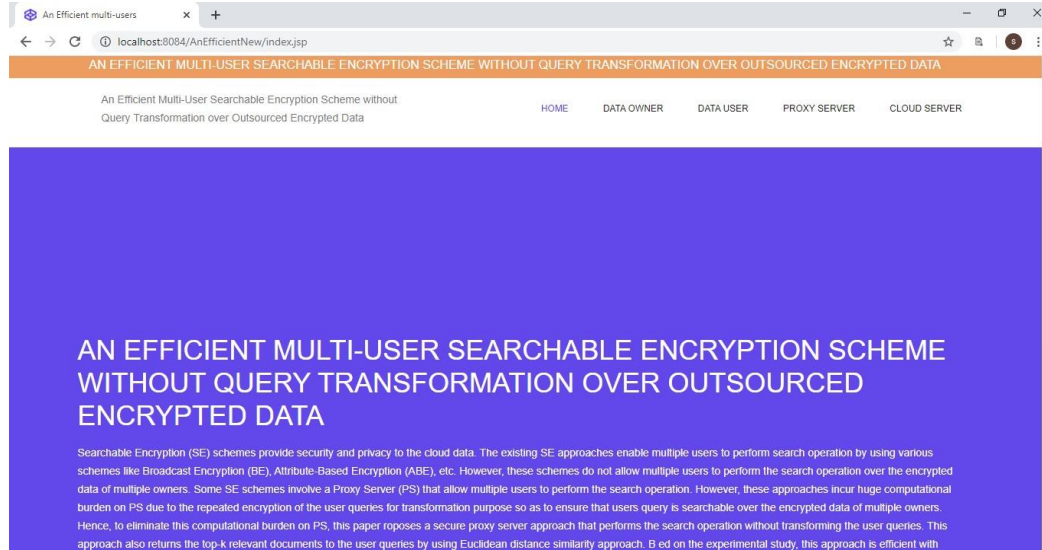


Fig 7.1 Home Page

This is the home page where description of "An efficient multi-user Searchable encryption scheme Without query transformation over outsourced encrypted data" had been described. The data Owner, data user, proxy server and cloud server can open their respite pages.

7.2 Data Owner Register

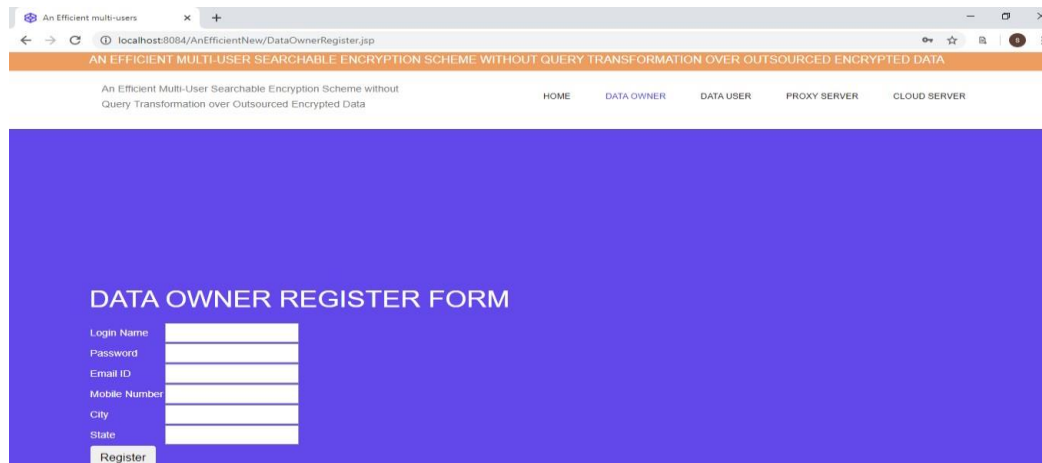
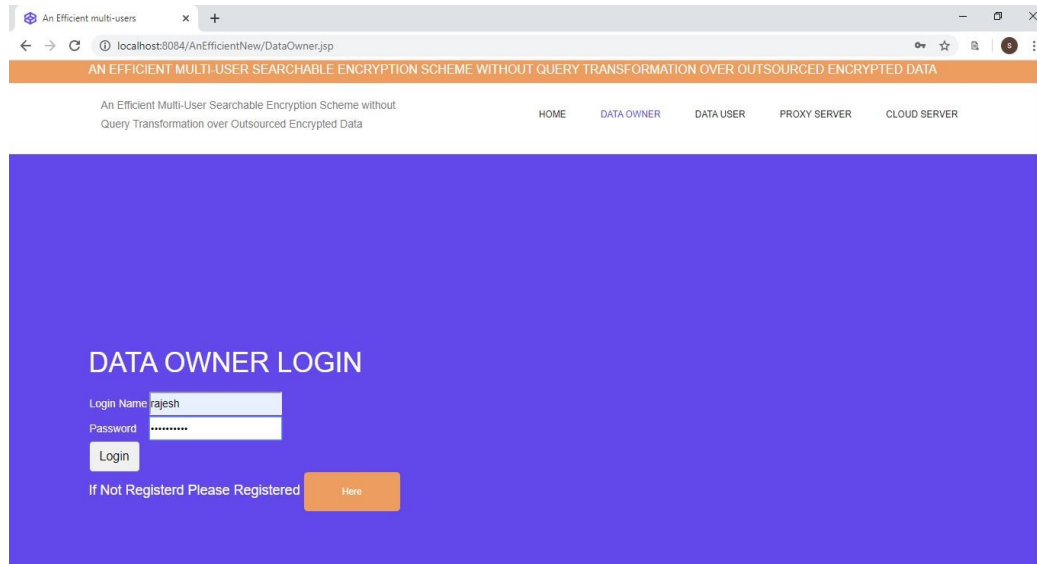


Fig 7.2 Data Owner Register

This is a page of data owner register by filling the above details. By registering this page Data owner can login and upload files.

7.3 Data Owner Login Page



An Efficient multi-users

localhost:8084/AnEfficientNew/DataOwner.jsp

AN EFFICIENT MULTI-USER SEARCHABLE ENCRYPTION SCHEME WITHOUT QUERY TRANSFORMATION OVER OUTSOURCED ENCRYPTED DATA

An Efficient Multi-User Searchable Encryption Scheme without Query Transformation over Outsourced Encrypted Data

HOME DATA OWNER DATA USER PROXY SERVER CLOUD SERVER

DATA OWNER LOGIN

Login Name: rajesh

Password:

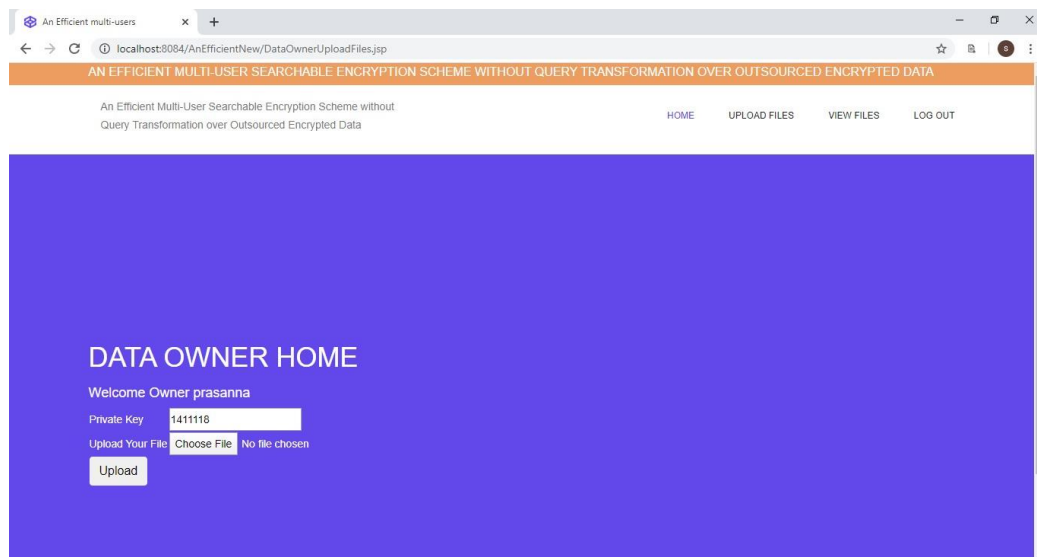
Login

If Not Registered Please Registered [Here](#)

Fig 7.3 Data owner Login page

After successful registering only the owners can login to this page where data owner can login by filling the respective details. By login to the respective page data owner can upload data files. If owner enters wrong login name or password, it will not proceed to the next page to upload a file. If in case owner does not register then he/she can register by clicking on 'Here'.

7.4 Data Owner Upload Files



An Efficient multi-users

localhost:8084/AnEfficientNew/DataOwnerUploadFiles.jsp

AN EFFICIENT MULTI-USER SEARCHABLE ENCRYPTION SCHEME WITHOUT QUERY TRANSFORMATION OVER OUTSOURCED ENCRYPTED DATA

An Efficient Multi-User Searchable Encryption Scheme without Query Transformation over Outsourced Encrypted Data

HOME UPLOAD FILES VIEW FILES LOG OUT

DATA OWNER HOME

Welcome Owner prasanna

Private Key: 1411118

Upload Your File: [Choose File](#) No file chosen

Upload

Fig 7.4 Data owner Upload Files

This is a page where data owner can upload by choosing the file.

7.5 Data Owner Upload File Data Indexes:

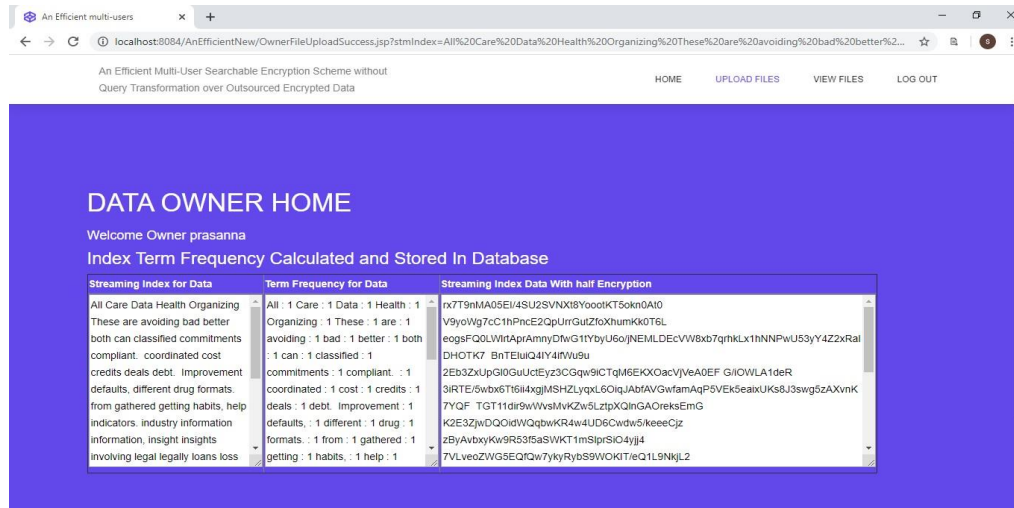


Fig 7.5 Data Owner Upload File data Indexes

This is a page where the upload file by data owner is half encrypted and send to proxy server.

7.6 Data Owner File Waiting If Proxy Server Accepts Then He/She Can Download

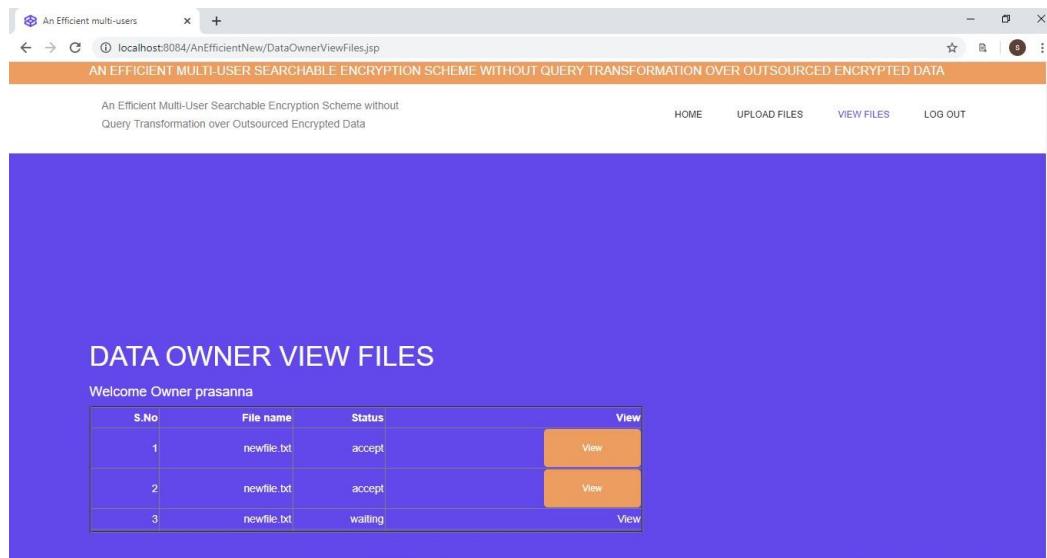


Fig 7.6 Data owner File Waiting if Proxy Server accepts then he/she can download

Half encrypted data file document has to accept by proxy server for that purpose. Here data Owner is waiting. After accepting only data owner can view the data.

7.7 Proxy Server Login Page

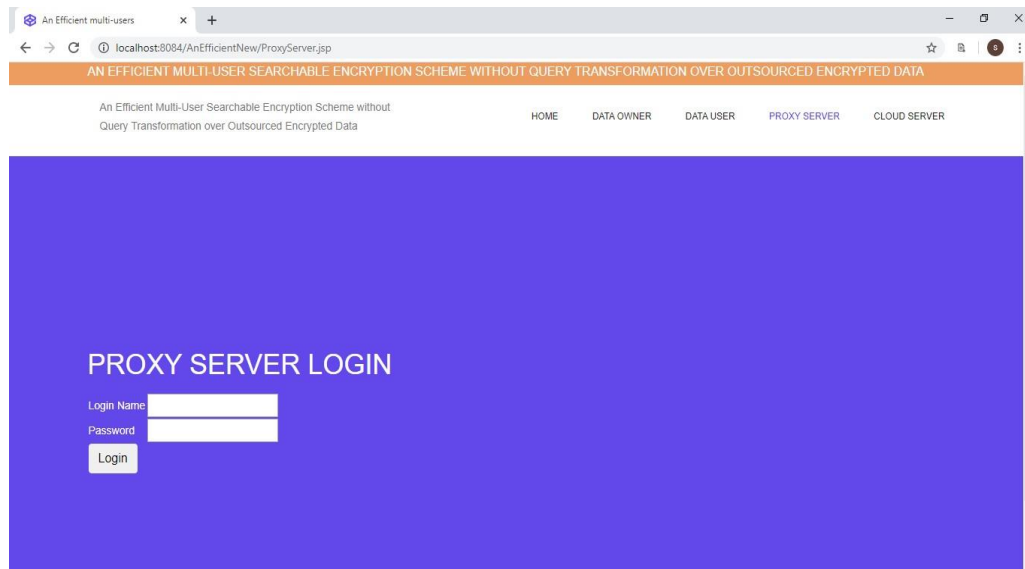


Fig 7.7 Proxy Server Login Page

This is page where administrator of proxy server can login by giving the name and password.

7.8 Proxy User Home Page

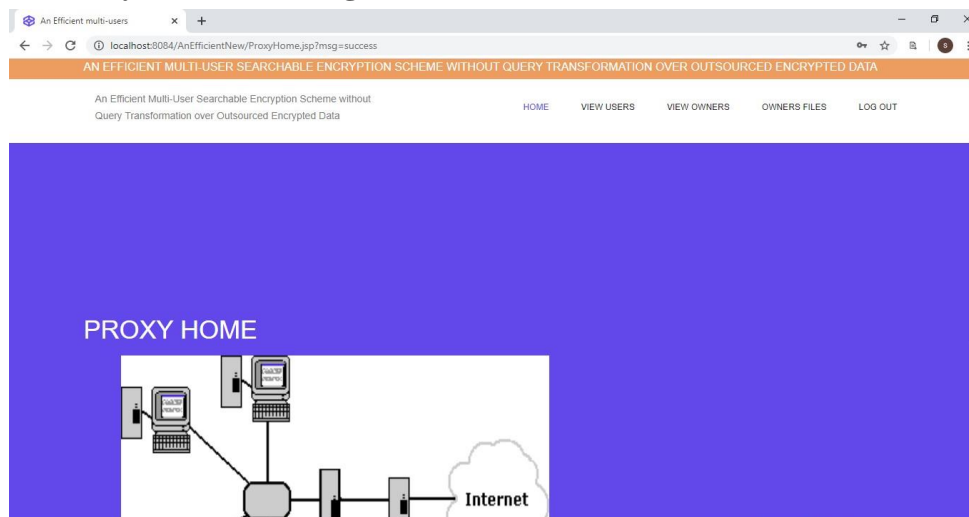


Fig 7.8 Proxy User Home Page

This is a home page of proxy server administrator where proxy server can view users, owners and Owners uploaded documents.

7.9 Proxy View Registered Owners

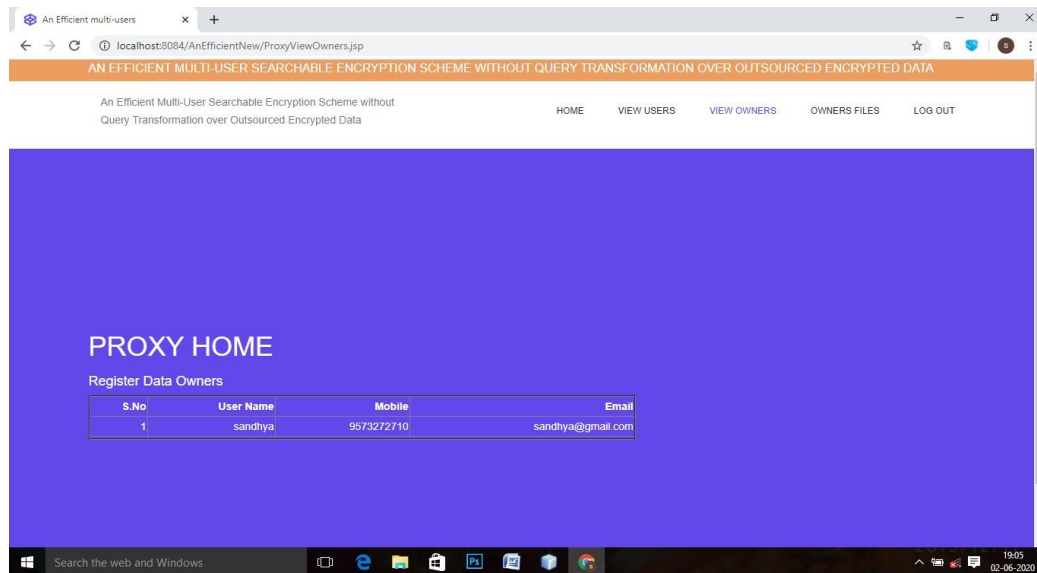


Fig 7.9 Proxy View Registered Owners

This is the page which proxy user gets by clicking on the view the owners who have registered.

7.10 View Registered Data Users

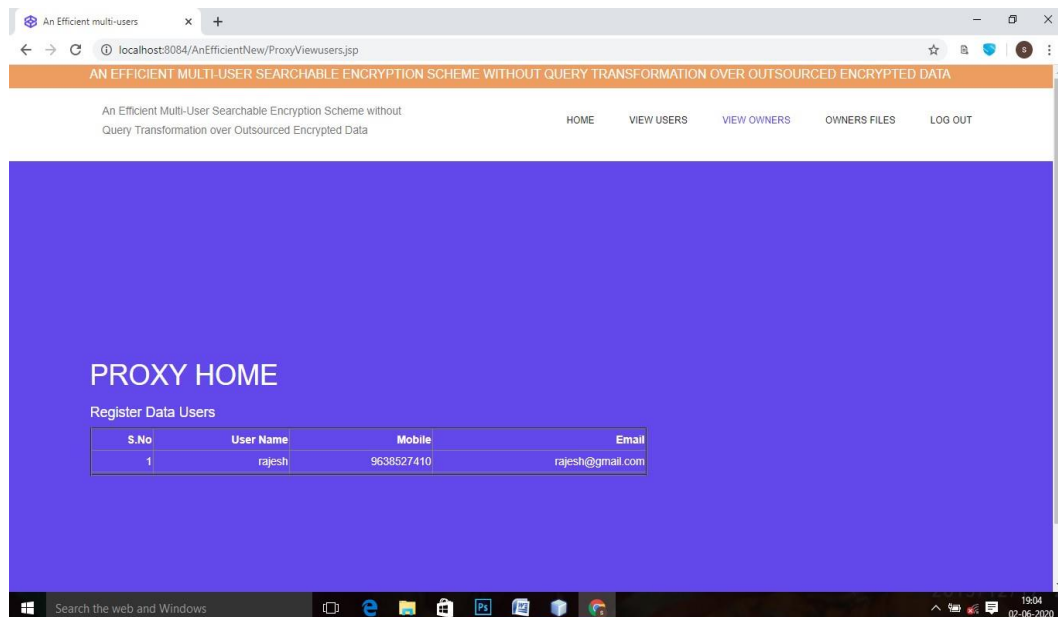


Fig 7.10 View Registered Data Users

This is the page which proxy user gets by clicking on view users. Here we view the users Who have registre.

7.11 Data owners File View

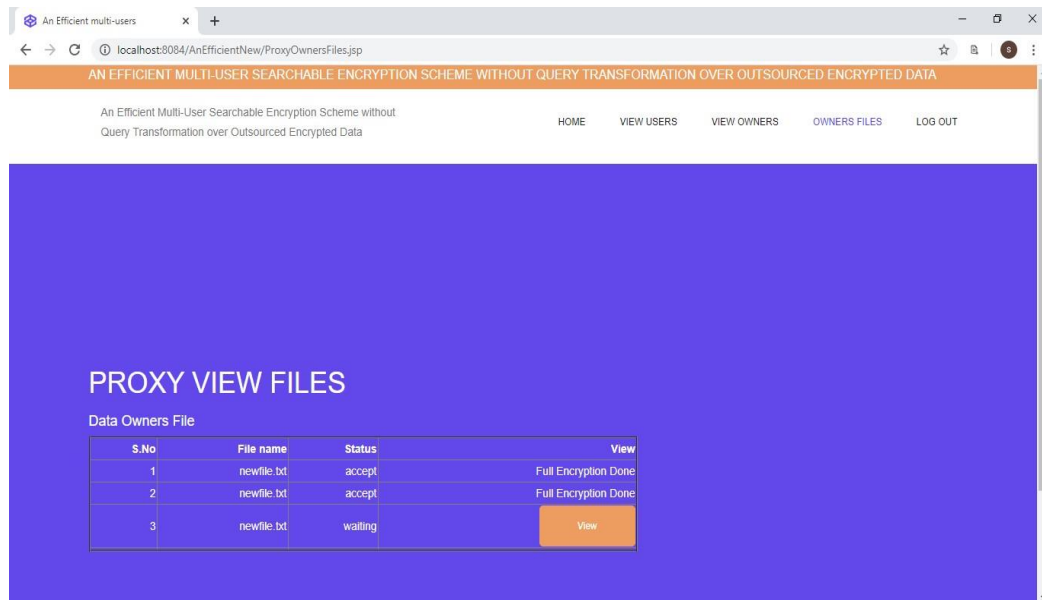


Fig 7.11 Data owners File View

This is a page where proxy user gets by clicking on the "owners files". Here proxy user accept and fully encrypt the file which uploaded by data owners.

7.12 Data user File half Encryption View

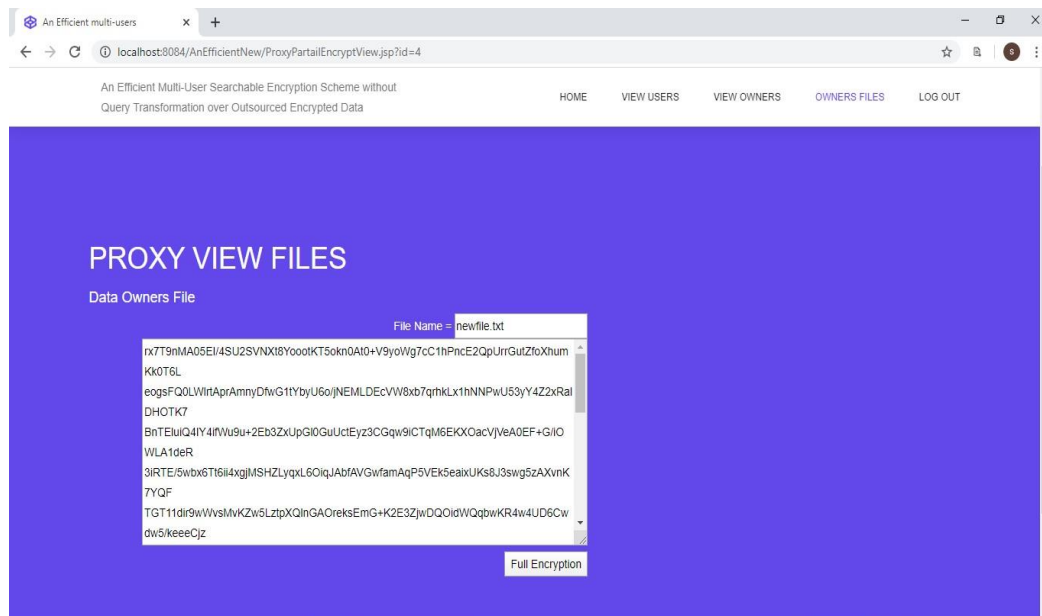


Fig 7.12 Data user File half Encryption View

Here half encrypted file will converted into fully encrypted form.

7.13 Data owner File Full Encryption Done

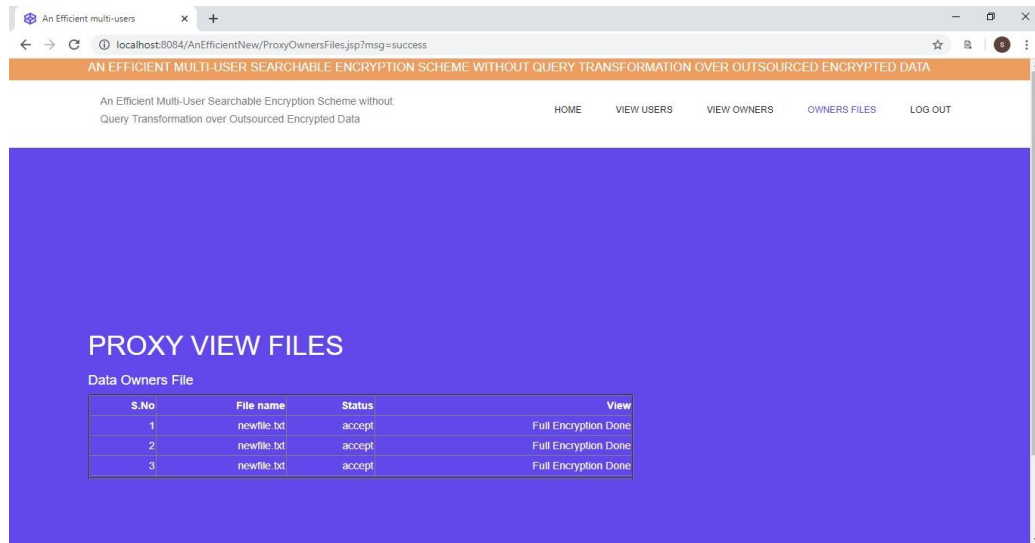


Fig 7.13 Data owner File Full Encryption Done

After fully encryption those files are viewed here.

7.14 Data Users Register Form:

The screenshot shows a web browser window with the URL `localhost:8084/AnEfficientNew/DataUserRegister.jsp`. The page has a blue header with the title "AN EFFICIENT MULTI-USER SEARCHABLE ENCRYPTION SCHEME WITHOUT QUERY TRANSFORMATION OVER OUTSOURCED ENCRYPTED DATA". Below the header, there is a navigation bar with links: HOME, DATA OWNER, DATA USER, PROXY SERVER, and CLOUD SERVER. The main content area has a blue background and displays the title "DATA USER REGISTER FORM". Below this title, there is a registration form with the following fields:

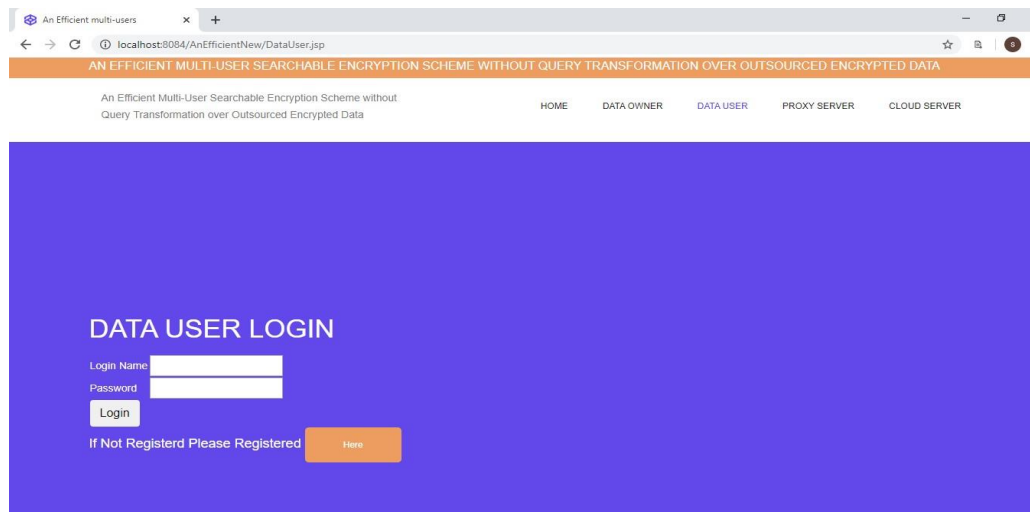
- Login Name
- Password
- Email ID
- Mobile Number
- City
- State

At the bottom of the form, there is a "Register" button.

Fig 7.14 Data Users Register Form

After the logout from proxy server by proxy user, this page is get by clicking on data user. This page is for registration purpose. Datauser registers by filling the above details.

7.15 Data user Login:



AN EFFICIENT MULTI-USER SEARCHABLE ENCRYPTION SCHEME WITHOUT QUERY TRANSFORMATION OVER OUTSOURCED ENCRYPTED DATA

An Efficient Multi-User Searchable Encryption Scheme without Query Transformation over Outsourced Encrypted Data

HOME DATA OWNER DATA USER PROXY SERVER CLOUD SERVER

DATA USER LOGIN

Login Name:

Password:

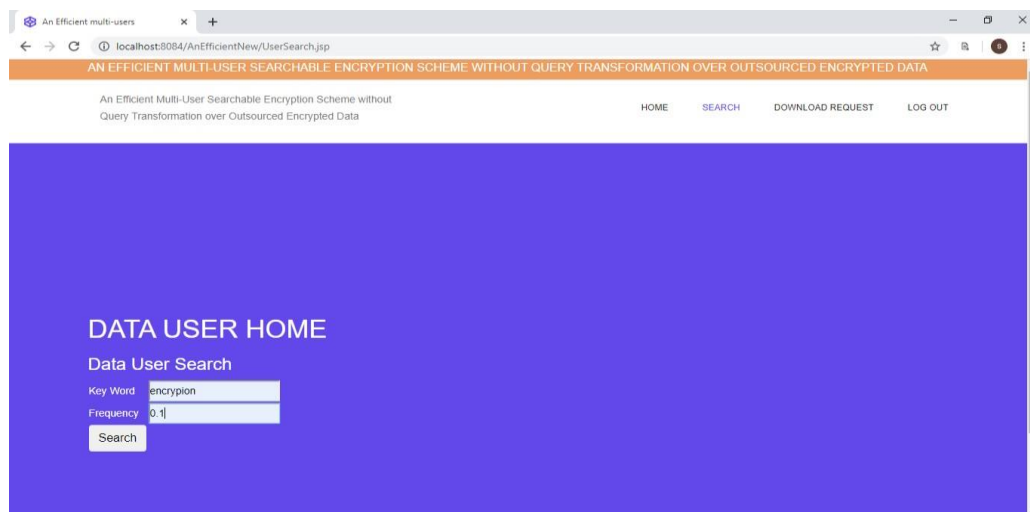
Login

If Not Registered Please Registered [Here](#)

Fig 7.15 Data user Login

This is the page where data user can login by giving name and password. If user did not Register then he/she have to click on 'Here' to register.

7.16 Data user Search:



AN EFFICIENT MULTI-USER SEARCHABLE ENCRYPTION SCHEME WITHOUT QUERY TRANSFORMATION OVER OUTSOURCED ENCRYPTED DATA

An Efficient Multi-User Searchable Encryption Scheme without Query Transformation over Outsourced Encrypted Data

HOME SEARCH DOWNLOAD REQUEST LOG OUT

DATA USER HOME

Data User Search

Key Word:

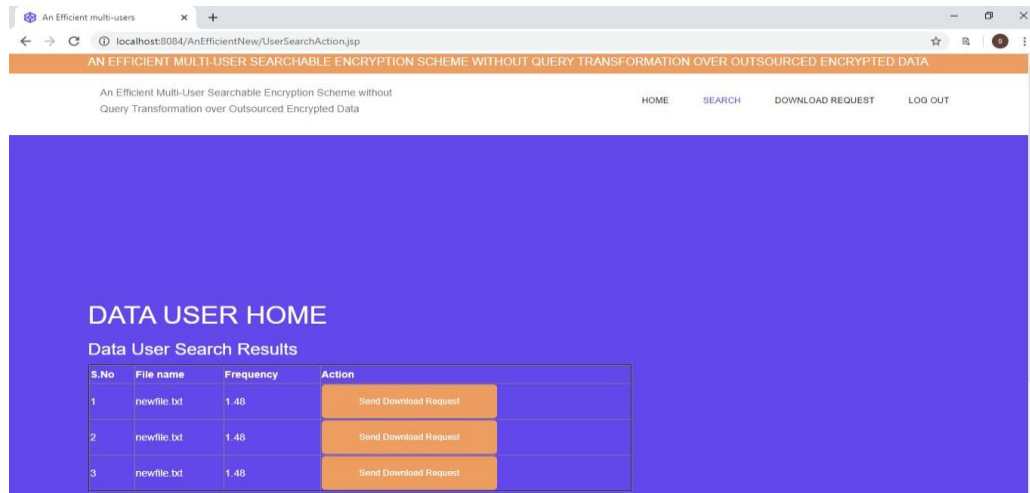
Frequency:

Search

Fig 7.16 Data user Search

This is the page which execute after login is successfully done by user. Here user has to Give the keyword and its frequency and click on "search". According to the given Keyword and given frequency search operation performed.

7.17 Data user Search Result:



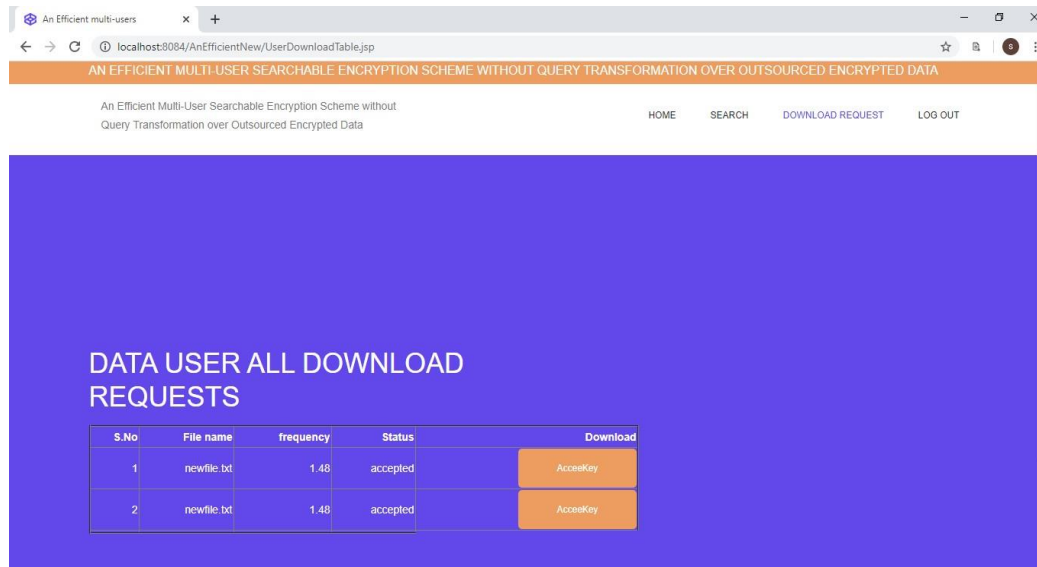
The screenshot shows a web browser window with the URL `localhost:8084/AnEfficientNew/UserSearchAction.jsp`. The page has a blue header with the title "AN EFFICIENT MULTI-USER SEARCHABLE ENCRYPTION SCHEME WITHOUT QUERY TRANSFORMATION OVER OUTSOURCED ENCRYPTED DATA" and navigation links: HOME, SEARCH, DOWNLOAD REQUEST, and LOG OUT. The main content area is blue and contains the heading "DATA USER HOME" and "Data User Search Results". Below this is a table with 4 columns: S.No, File name, Frequency, and Action. The table contains 3 rows of data, each with a "Send Download Request" button in the Action column.

| S.No | File name | Frequency | Action |
|------|-------------|-----------|--|
| 1 | newfile.txt | 1.48 | <button>Send Download Request</button> |
| 2 | newfile.txt | 1.48 | <button>Send Download Request</button> |
| 3 | newfile.txt | 1.48 | <button>Send Download Request</button> |

Fig 7.17 Data user Search Result

After performing search operation, the following resulted files are executed. The required files the user can access after sending the request.

7.18 Download Request Sent:



The screenshot shows a web browser window with the URL `localhost:8084/AnEfficientNew/UserDownloadTable.jsp`. The page has a blue header with the title "AN EFFICIENT MULTI-USER SEARCHABLE ENCRYPTION SCHEME WITHOUT QUERY TRANSFORMATION OVER OUTSOURCED ENCRYPTED DATA" and navigation links: HOME, SEARCH, DOWNLOAD REQUEST, and LOG OUT. The main content area is blue and contains the heading "DATA USER ALL DOWNLOAD REQUESTS". Below this is a table with 6 columns: S.No, File name, frequency, Status, and Download. The table contains 2 rows of data, each with an "AccessKey" button in the Download column.

| S.No | File name | frequency | Status | Download |
|------|-------------|-----------|----------|----------------------------|
| 1 | newfile.txt | 1.48 | accepted | <button>AccessKey</button> |
| 2 | newfile.txt | 1.48 | accepted | <button>AccessKey</button> |

Fig 7.18 Download Request Sent

This page is execute by clicking on 'download request'. Here the user accepted file are shown which can be accessed by access key which has been send to the user mail. After entering the access key only user can able to access file.

8. CONCLUSION

A Proxy server based approach for supporting search operation over the data of multiple owners is proposed. Different from the existing approaches, the data user's query in this approach can be used to search over the multiple owners' data without transforming the query. In order to bypass the query transformation, the idea of partial encryption is used, i.e., half of each of the both index keyword and query keyword are encrypted by using the secret key of the data owner and the data user respectively and the other half of the index keyword and query keyword is encrypted by using common secret key of the proxy server. The experimental results confirm that the proposed approach is efficient. Future work could be to include a module for addition and revocation of data users and also to enhance the security functionalities of the proposed approach

REFERENCES

- [1] D. X. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in *Security and Privacy*, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on. IEEE, 2000, pp. 44–55.
- [2] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2004, pp. 506–522.
- [3] J. Lotspiech, “12 - broadcast encryption,” in *Multimedia Security Technologies for Digital Rights Management*, W. Zeng, H. Yu, and C.-Y. Lin, Eds. Burlington: Academic Press, 2006, pp. 303 – 322.
- [4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS ’06. New York, NY, USA: ACM, 2006, pp. 89–98.
- [5] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, “Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing,” *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1566–1577, 2016.
- [6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: improved definitions and efficient constructions,” *CCS-2006:ACM conference on Computers and Communications Security*, pp. 79–88, 2006.
- [7] Q. Wang, Y. Zhu, and X. Luo, “Multi-user searchable encryption with fine-grained access control without key sharing,” in *2014 3rd International*

Conference on Advanced Computer Science Applications and Technologies, Dec 2014, pp. 145–150.

[8] Z. Deng, K. Li, K. Li, and J. Zhou, “A multi-user searchable encryption scheme with keyword authorization in a cloud storage,” *Future Generation Computer Systems*, vol. 72, pp. 208–218, 2017.

[9] T. Korenius, J. Laurikkala, and M. Juhola, “On principal component analysis, cosine and Euclidean measures in information retrieval,” *Information Sciences*, vol. 177, no. 22, pp. 4893 – 4905, 2007.

[10] RFC, “Request for comments database,” <https://www.rfceditor.org/retrieve/bulk/>.