A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION

A PROJECT REPORT

Submitted by

THARUNESHWARAN G

211420104288

VENGATESWARAN R

211420104299

VIMAL M

211420104305

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION "** is the bonafide work of **" THARUNESHWARAN G (211420104288) , VENGATESWARAN R (211420104299) , VIMAL M (211420104305) "** who carried out the project work under my supervision.

Signature of the HOD with date	**Signature of the Supervisor with date**
Dr. L.JABASHEELA M.E., Ph.D.,	**Mr. C.THYAGARAJAN M.E., (Ph.D.)**
PROFESSOR AND HEAD ,	**SUPERVISOR ,**
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Panimalar Engineering College,	Panimalar Engineering College,
Chennai-123.	Chennai-123.

Certified that the above candidate(s) was examined in the End Semester Project Viva - Voce Examination held on ____________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We THARUNESHWARAN.G (211420104288), VENGATESWARAN.R (211420104299), VIMAL.M (211420104305) hereby declare that this project report titled "**A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION**", under the guidance of Mr. C. THYAGARAJAN M.E., (Ph.D.) is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

THARUNESHWARAN.G[211420104288]

VENGATESWARAN.R[211420104299]

VIMAL.M[211420104305]

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Dr. V. SUBEDHA M.Tech., Ph.D.,** and **Mr. C. THYAGARAJAN M.E., (Ph.D.)** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

THARUNESHWARAN.G[211420104288]

VENGATESWARAN.R[211420104299]

VIMAL.M[211420104305]

ABSTRACT

Cloud storage services have emerged as indispensable components of the rapidly expanding landscape of cloud computing, offering unparalleled convenience and scalability. Despite their widespread adoption and utility, cloud storage platforms face significant security challenges stemming from management oversights and malicious attacks. These vulnerabilities oftenresult in extensive breaches and the leakage of sensitive data. To address these concerns, this project proposes Cloud Secure Storage Mechanism (CSSM) designed to fortify the confidentiality of data stored in the cloud.One of its key strategies involves the integration of data dispersion anddistributed storage techniques. By dispersing data into fragments and distributing them across disparate storage locations, Cloud Secure Storage Mechanism mitigates the risk of data compromise and loss. Furthermore, the mechanism employs encryption, chunking, and distributed storage methodologies to safeguard data integrity and confidentiality. And to access these dispersed data we search them by their mapped keywords. This encryption scheme ensures that even if individual data fragments are compromised, the overall security of the system remains intact. Cloud Secure Storage Mechanism also adopts a hierarchical management approach to streamline resource allocation and access control. This hierarchical structure facilitates efficient management of cloud storage resources, optimizing storage utilization and enhancing administrative oversight.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Cloud computing has evolved significantly, playing a vital role in applications like pattern recognition, image forensic, and forgery detection. At the forefront of this evolution is Amazon Web Service (AWS), serving as the de facto standard in the cloud industry. OpenStack Swift, a core component of the OpenStack framework aligned with AWS standards, has gained widespread popularity as a cloud storage mechanism. However, despite its ubiquity, security challenges persist within OpenStack Swift. The default configuration often results in the storage of data in plaintext, posing risks of unauthorized access and data leakage at the storage layer. The Cloud Security Alliance's analysis report highlights user data leakage as a major concern, with management negligence and malicious attacks accounting for a significant portion of reported cloud security cases. In response to these challenges, this project introduces the Cloud Secure Storage Mechanism (CSSM). Cloud Secure Storage Mechanism addresses security vulnerabilities in OpenStack Swift by implementing a combination of data dispersion and encryption. This innovative approach ensures that large-scale cloud data and cryptographic keys are stored in chunked cipher texts, effectively mitigating the risk of unauthorized access. To enhance key security further, Cloud Secure Storage Mechanism incorporates user password protection and secret sharing mechanisms. The significance of this work lies in its potential to offer a comprehensive solution for preventing data leakage at the storage layer in cloud environments. By addressing the vulnerabilities associated with default configurations and introducing robust security measures, Cloud Secure Storage Mechanism contributes to the continuous efforts aimed at enhancing the overall security of cloud storage systems. It’s holistic approach aligns with the evolving landscape of cloud computing, providing a valuable contribution to the ongoing discourse on securing sensitive data in cloud environments. Cloud Secure Storage Mechanism combines data dispersion withdata encryption, so that large-scale cloud data and keys would be stored in chunked cipher texts. On this basis, user password and secret sharing are introduced to further protect keys security.

We implemented Cloud Secure Storage Mechanism based on OpenStack Swift mechanism and made several tests. The major contributions of this work are listed below:

1) **Data Secure Storage:** In order to prevent data leakage and increase the difficulty of attack, this project presents a method combining data distribution and data encryption to improve data storage security.

2) **Key Management:** To protect the key and prevent the attacker from using the key to recover the data, this project introduces secret sharing to enhance key security.

Research in cloud computing is receiving a lot of attention from both academic and industrial worlds. In cloud computing, users can outsource their computation and storage to servers (also called clouds) using Internet. Clouds can provide several types of services like applications (e.g., Google Apps, Microsoft online), infrastructures (e.g., Amazon’s EC2, Eucalyptus, Nimbus), and platforms to help developers write applications (e.g., Amazon’s S3, Windows Azure). Much of the data stored in clouds is highly sensitive, for example, medical records and social networks. Security and privacy are thus very important issues in cloud computing. In one hand, the user should authenticate itself before initiating any transaction, and on the other hand, it must be ensured that the cloud does not tamper with the data that is outsourced.

1.2 PROBLEM DEFINITION

Cloud computing has rapidly evolved in recent years, with storage as a service becoming a central component for various applications like pattern recognition and image forensics. Amazon Web Services (AWS) has established itselfas the industry standard, with Swift, an integral part of OpenStack, emerging as a popular cloud storage mechanism. However, ensuring data security at the storage layer remains a challenge. Outsourced encrypted Data are directly stored in cloud, which may lead to severe confidentiality and privacy issues. Searchable encryption schemes fail to offer sufficient insights towards the construction of full functioned search over encrypted cloud data. Bulk content retrieval for file searching, which is inefficient.

CHAPTER 2
LITERATURE REVIEW

1. **Author:** Geeta Sharma, Sheetal Kalra

Title: "A Lightweight User Authentication Scheme for Cloud-IoT Based Healthcare Services,"

Description: With the ongoing revolution of cloud computing and Internet of Things, remote patient monitoring has become feasible. These networking paradigms are widely used to provide healthcare services and real-time patient monitoring. The sensors that are either wearable or embedded within the body of a patient transmit patient's data to the remote medical centers. The medical professional can access patient's data stored in the cloud anywhere across the globe. As the sensitive data of the patient are sent over insecure cloud-IoT networks, secure user authentication is of utmost importance. An efficient user authentication scheme ensures that only legitimate users can access data and services. This paper proposes a secure and efficient user authentication scheme for remote patient monitoring. The proposed scheme is robust, lightweight and secure against multiple security attacks. Furthermore, the scheme has low computational overhead. A formal verification using AVISPA tool confirms the security of the proposed scheme.

Advantages:

1. Enhanced security tailored for Cloud-IoT healthcare.

2. Efficiency with minimal computational overhead.

3. Scalability for growing IoT device and user base.

4. User-friendly experience for healthcare professionals and patients.

5. Compatibility with existing Cloud-IoT infrastructures.

Disadvantages:

1. Potential security trade-offs compared to robust authentication.

2. Limited features due to lightweight nature.

3. Vulnerability to specific security risks.

4. Dependence on reliable network connectivity.

5. Challenges in adoption and integration into existing systems.

2. Author: Geeta Sharma, Sheetal Kalra 2019

Title: Advanced Lightweight Multi-Factor Remote User Authentication Scheme for Cloud-IoT Applications,”

Description: With the ongoing revolution of Internet-enabled devices, Internet of Things (IoT) has emerged as the most popular networking paradigm. The enormous amount of data generated from smart devices in IoT environment is one of the biggest concerns. Cloud computing has emerged as a key technology to process the generated data. The confidential data of user from IoT devices is stored in cloud server and the remote user can access this data anytime, anywhere and at any place from the cloud server. This makes remote user authentication a critical issue. This paper proposes a lightweight remote user authentication scheme for cloud-IoT applications. The formal security analysis using BAN logic and random oracle model confirms that the scheme is resilient to known security attacks. Furthermore, the scheme is formally verified using AVISPA tool which confirms the security against multiple security attacks.

Advantages:

1. Enhanced Security: Provides robust multi-factor authentication for Cloud-IoT applications.

2. Efficiency: Maintains efficiency with minimal resource usage.

3. Scalability: Easily integrates with expanding IoT networks.

4. Usability: Offers a user-friendly authentication process.

5. Adaptability: Compatible with diverse Cloud-IoT architectures.

Disadvantages:

1. Complexity: Requires careful configuration and management.

2. Resource Consumption: May impact device performance.

3. Integration Challenges: Compatibility issues may arise.

4. Potential Vulnerabilities: New attack vectors may emerge.

5. User Training: Users may require education on the authentication process.

3. **Author:** Chandrakar P, Om H 2019

Title: An efficient two-factor remote user authentication and session key agreement scheme using rabin cryptosystem

Description: In this paper, we present a safe and reliable remote user authentication and session key agreement scheme using the Rabin cryptosystem. The Rabin cryptosystem relies on prime integer factorization, which provides high security. Our scheme is validated using the Burrows–Abadi–Needham logic, which proved that it facilitates mutual authentication and session key agreement securely. The informal security analysis shows that the proposed scheme is secured against various malicious attacks. We simulate our scheme using the well-known Automated Validation of Internet Security Protocols and Applications tool, which confirms that the proposed scheme can defend from the passive and active attacks and also prevents the man-in-the-middle and replay attacks. Further, the performance evaluation shows that our scheme provides robustness against various security attacks as well as efficient in the terms of smart card storage cost, estimated execution time, communication cost and computation cost.

Advantages:

1. Enhanced Security: Two-factor authentication adds extra security.
2. Efficiency: Uses Rabin cryptosystem for efficient cryptographic operations.
3. User-Friendly: Offers an efficient and user-friendly authentication process.
4. Session Key Agreement: Facilitates secure session key agreement.
5. Resistance to Attacks: Provides resistance against cryptographic attacks.

Disadvantages:

1. Key Management: Requires effective key management.
2. Implementation Complexity: May involve complex implementation efforts.
3. Compatibility: Compatibility issues may arise during integration.
4. Vulnerabilities: Potential vulnerabilities may exist despite robustness.
5. Scalability: Efficiency may degrade with a large number of users.

4. **Author:** Yoon, E.-J.; Ryu, E.-K.; Yoo, K.-Y 2004

Title: Further improvement of an efficient password based remote user authentication scheme using smart cards

Description: Recently, Ku-Chen proposed an improvement to Chien et al.'s scheme to prevent from some weaknesses. However, the improved scheme is not only still susceptible to parallel session attack, but also insecure for changing the user's password in password change phase. Accordingly, the current paper presents an enhancement to resolve such problems. As a result, the proposed scheme enables users to change their passwords freely and securely without the help of a remote server, while also providing secure mutual authentication.

Advantages:

1. Enhanced Security: Strengthens password-based authentication with smart cards.

2. Two-Factor Authentication: Adds an extra layer of security.

3. Resistance to Attacks: Protects against common password attacks.

4. Efficient Process: Streamlines authentication for improved user experience.

5. Secure Key Exchange: Ensures confidentiality and integrity of communication.

Disadvantages:

1. Complexity: Integration may require expertise.

2. Smart Card Management: Requires effective management practices.

3. Compatibility: Integration may face compatibility issues.

4. Cost: Implementation may incur additional expenses.

5. Dependency: Relies on smart cards, which may pose challenges if lost or forgotten.

5. Author: H.Y. Chien, J. K. Jan and Y.M. Tseng 2002

Title: An efficient and practical solution to remote authentication: smart card

Description: The smart card-based scheme is a very promising and practical solution to remote authentication. Compared with other smart card-based schemes, our solution achieves more functionality and requires much less computational cost. These important merits include: (1) there is no verification table; (2) users can freely choose their passwords; (3) the communication cost and the computational cost is very low; and (4) it provides mutual authentication between the user and the
serve.

Advantages:

1. Enhanced Security: Utilizes smart cards for robust authentication.
2. Two-Factor Authentication: Adds an extra layer of security.
3. Resistance to Attacks: Protects against common authentication attacks.
4. Efficiency: Streamlines authentication for improved user experience.
5. Portability: Convenient authentication for remote users.

Disadvantages:

1. Complexity: Integration may require expertise.
2. Management: Requires effective smart card management.
3. Compatibility: Integration may face compatibility issues.
4. Cost: Implementation may incur additional expenses.
5. Dependency: Relies on smart cards, which may pose challenges if lost or forgotten.

6. **Author**: Yifei Chen, Meng Li, Shuli Zheng, Donghui Hu, Chhagan Lal & Mauro Conti

Title: One-Time, Oblivious, and Unlinkable Query Processing Over Encrypted Data on Cloud

Description: Searchable Symmetric Encryption aims at making possible searching over an encrypted database stored on an untrusted server while keeping privacy of both the queries and the data, by allowing some smallcontrolled leakage to the server. Recent work shows that dynamic schemes -- in which the data is efficiently updatable -- leaking some information on updated keywords are subject to devastating adaptative attacks breaking the privacy ofthe queries. The only way to thwart this attack is to design forward private schemes whose update procedure does not leak if a newly inserted element matches previous search queries. This work proposes Sophos as a forward private SSE scheme with performance similar to existing less secure schemes, and that is conceptually simpler (and also more efficient) than previous forward private constructions. In particular, it only relies on trapdoor permutations and does not use an ORAM-like construction. We also explain why Sophos is an optimal point of the security/performance trade-off for SSE.

Advantages:

1. Enhanced Privacy: Protects sensitive data with encryption.
2. One-Time Queries: Reduces query exposure for improved privacy.
3. Oblivious Processing: Prevents the cloud from learning about queries.
4. Unlinkable Queries: Ensures queries are not correlated with data or identities.
5. Cloud-Based: Utilizes cloud resources for scalability without compromising privacy.

Disadvantages:

1. Computational Overhead: May impact query performance.
2. Key Management: Requires effective management practices.
3. Limited Query Flexibility: One-time queries may restrict analysis options.
4. Complex Implementation: Integration may be challenging.
5. Scalability Challenges: Scaling for large datasets may be difficult.

7. **Author:** Xianrui Meng; Haohan Zhu; George Kollios

Title: Top-k Query Processing on Encrypted Databases with Strong Security Guarantees

Description: Concerns about privacy in outsourced cloud databases have grown recently and many efficient and scalable query processing methods over encrypted data have been proposed. However, there is very limited work on how to securely process top-k ranking queries over encrypted databases in the cloud. In this paper, we propose the first efficient and provably secure top-k query processing construction that achieves adaptive CQA security. We develop an encrypted data structure called EHL and describe several secure sub-protocols under our security model to answer top-k queries. Furthermore, we optimize our query algorithms for both space and time efficiency. Finally, we empirically evaluate our protocol using real world datasets and demonstrate that our construction is efficient and practical.

Advantages:

1. Data Privacy: Ensures confidentiality through encryption.
2. Strong Security: Provides robust security assurances.

3. Query Flexibility: Supports efficient retrieval of relevant information.
4. Scalability: Enables secure query processing on encrypted databases.
5. Privacy-Preserving: Safeguards sensitive data from unauthorized access.

Disadvantages:
1. Computational Overhead: May impact performance due to encryption.
2. Key Management: Requires effective management practices.
3. Limited Functionality: Encryption may restrict certain database operations.
4. Complex Implementation: Integration may be challenging.
5. Trade-offs: Balancing security and performance may involve compromises.

8. **Author**: Rui Li; Alex X. Liu

Title: Adaptively Secure Conjunctive Query Processing over Encrypted Data for Cloud Computing

Description: This paper concerns the fundamental problem of processing conjunctive queries that contain both keyword conditions and range conditions on public clouds in a privacy preserving manner. No prior Searchable Symmetric Encryption (SSE) based privacy-preserving conjunctive query processing scheme satisfies the three requirements of adaptive security, efficient query processing, and scalable index size. In this paper, we propose the first privacy preserving conjunctive query processing scheme that satisfies the above requirements. To achieve adaptive security, we propose an Indistinguishable Bloom Filter (IBF) data structure for indexing. To achieve efficient query processing and structure indistinguishability, we propose a highly balanced binary tree data structure called Indistinguishable Binary Tree (IBtree). To optimize searching efficiency, we propose a traversal width minimization algorithm and a traversal depth minimization algorithm. To achieve scalable andcompact index size, we propose an IBtree space compression algorithm to remove redundant information in IBFs. We formally prove that our scheme is adaptive secure using a random oracle model. The key contribution of this paper is on achieving conjunctive query processing with both strong privacy guarantee and practical efficiency in terms of both speed and space. We implemented our scheme in C++, evaluated and compared its performance with KRB [24] for keyword queries and PBtree [32] for range queries on two real-world data sets. Experimental results show that our scheme is fast and scalable (in milliseconds).

Advantages:

1. Data Privacy: Ensures confidentiality through encryption.
2. Adaptive Security: Provides robust protection against threats.
3. Efficient Query Processing: Maintains query functionality while preserving security.
4. Scalability: Supports query processing in cloud environments.
5. Privacy-Preserving: Prevents unauthorized access to sensitive data.

Disadvantages:

1. Computational Overhead: May impact performance due to encryption.
2. Key Management: Requires effective management practices.
3. Complexity: Integration may be complex.
4. Limited Functionality: Encryption may restrict some operations.
5. Trade-offs: Balancing security with performance may involve compromises.

9. **Author**: u Lei; Alex X. Liu; Rui Li

Title: Secure KNN Queries over Encrypted Data: Dimensionality Is Not Always a Curse

Description: The fast-increasing location-dependent applications in mobile devices are manufacturing a plethora of geospatial data. Outsourcing geospatial data storage to a powerful cloud is an economical approach. However, safeguarding data users' location privacy against the untrusted cloud while providing efficient location-aware query processing over encrypted data are in conflict with each other. As a step to reconcile such conflict, we study secure k nearest neighbor (SkNN) queries processing over encrypted geospatial data in cloud computing. We design 2D SkNN (2DSkNN), a scheme achieves both strong provable security and high-efficiency. Our approach employs locality sensitive hashing (LSH) in a dimensional-increased manner. This is a counter- intuitive leverage of LSH since the traditional usage of LSH is to reducethe data dimensionality and solve the so-called "curse of dimensionality" problem. We show that increasing the data dimensionality via LSH is indeed helpful to tackle 2DSkNN problem. By LSH-based neighbor region encoding and two-tier prefix-free encoding, we turn the proximity test to be sequential keywords query with a stop condition, which can be well addressed by any existing symmetric searchable encryption (SSE) scheme. We show that 2DSkNNachieves adaptive indistinguishability under chosen-keyword attack (IND2-CKA) secure in the random oracle model. A prototype implementation and experiments on both real-world and synthetic datasets confirm the high practicality of 2DSkNN.

Advantages:

1. Data Privacy: Protects sensitive data through encryption.
2. Secure KNN Queries: Ensures privacy during k-nearest neighbor (KNN) query processing.
3. Efficiency: Maintains efficiency even with high-dimensional data.
4. Scalability: Supports KNN queries in large-scale data environments.
5. Privacy-Preserving: Prevents unauthorized access to sensitive data.

Disadvantages:

1. Computational Overhead: May impact query performance.
2. Key Management: Requires effective encryption key management.
3. Complexity: Integration may be complex.
4. Trade-offs: Balancing security and performance may involve compromises.
5. Limited Functionality: Encryption may restrict some database operations.

10. **Author:** Xinyu Lei; Alex X. Liu; Rui Li; Guan-Hua Tu

Title: SecEQP: A Secure and Efficient Scheme for SkNN Query Problem Over Encrypted Geodata on Cloud

Description: Nowadays, location-based services are proliferating and being widely deployed. For example, a Yelp user can obtain a list of the recommended restaurants near his/her current location. For some small or medium location service providers, they may rely on commercial cloud services, e.g., Dropbox, to store the tremendous geospatial data and deal with a number of user queries. However, it is challenging to achieve a secure and efficient location-based query processing over encrypted geospatial data stored on the cloud. In this paper, we propose the Secure and Efficient Query Processing (SecEQP) scheme to address the secure k nearest neighbor (SkNN) query problem. SecEQP employs the projection function-based approach to code neighbor regions of a given location. Given the codes of two locations, the cloud server only needs to comparewhether codes equal or not to check the proximity of the two locations. The codes are further embedded into an indistinguishable Bloom filter tree to build a secure and efficient index. The security of SecEQP is formally proved in the random oracle model. We further prototype SecEQP scheme and evaluate its performance on both real-world and synthetic datasets. Our evaluation results show that SecEQP is a highly efficient approach, e.g., top-10 NN query over 1 million datasets only needs less than 40 msec to get queried results

Advantages:

1. Data Privacy: Protects geodata confidentiality through encryption.
2. Secure SkNN Queries: Ensures privacy in spatial k-nearest neighbor (SkNN) queries.
3. Efficiency: Provides a fast solution for SkNN query processing.
4. Scalability: Supports SkNN queries in cloud environments for large-scale analysis.
5. Privacy-Preserving: Safeguards sensitive geodata from unauthorized access.

Disadvantages:

1. Computational Overhead: May impact query performance.
2. Key Management: Requires effective encryption key management.
3. Complexity: Integration may be complex.
4. Trade-offs: Balancing security and performance may involve compromises.
5. Limited Functionality: Encryption may restrict certain database operations.

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing system of our project highlights the rapid evolution of cloud computing, particularly in storage services, which have become essential for applications like pattern recognition and image forensics. Amazon Web Services (AWS) has solidified its position as the industry standard, with Swift, a key component of OpenStack, gaining popularity as a cloud storage mechanism. Despite these advancements, ensuring data security at the storage layer remains a challenge such as managing encryption keys, implementing access controls, ensuring compliance across locations, verifying data integrity, preventing insider threats, and securing data during transit. These require robust encryption, access controls, compliance management, data integrity checks, monitoring, and ongoing risk assessments. Encrypted data outsourced to the cloud directly may pose significant confidentiality and privacy risks. Existing searchable encryption schemes fall short inproviding comprehensive solutions for fully functional search capabilities over encrypted cloud data.

Disadvantages

- Requires more storage space and network bandwidth than traditionalcloud storage methods, as it stores encrypted and dispersed data chunks.
- Confidentiality and Privacy Risks: Storing encrypted data directly in the cloud poses risks to confidentiality and privacy, as unauthorized access or breaches could compromise sensitive information.
- Inefficient Search Capabilities: Existing searchable encryption schemes fail to provide comprehensive solutions for efficient and fully functional search capabilities over encrypted cloud data. This inefficiency hampers the effectiveness of data retrieval and search operations.

3.2 PROPOSED SYSTEM

The Cloud Secure Storage Mechanism (CSSM) proposed in this project aims to prevent data breaches at the storage layer within cloud environments. By integrating data dispersion and distributed storage, CSSM achieves encrypted, chunked, and distributed storage, enhancing the security of cloud storage systems. Additionally, CSSM adopts a hierarchical management approach and combines user passwords with secret sharing to prevent cryptographic material leakage, ensuring the confidentiality of sensitive data. We introduced an efficient and reliable methodology for search over encrypted data. Here the encrypted keyword search pre computes the resulting search documents for the input query from users through Natural language processing Technique which is implemented on gateway (client side) on user file upload. Overall, CSSM represents a significant advancement in cloud storage security, offering a robust solution to safeguard against data breaches. Through its innovative techniques and comprehensive approach, CSSM provides highly secure storage for cloud environments, minimizing the risk of unauthorized access to sensitive data.

Advantages

- The data which is stored in the cloud that is very safe and secure.
- There is no leakage of data.
- We can share the data with the secret key, trapdoor and the authentication is strong.
- To be able to search files using the keywords inside the content of the file.

3.3 REQUIREMENT ANALYSIS AND SPECIFICATION

➢ **Hardware Requirements**

- Cloud Server
- Laptop (Intel i3 processor)
- Internet connection

➢ **Software Requirements**

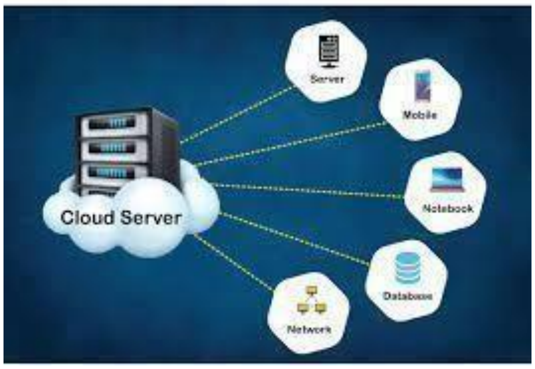
- Java Programming Language
- Java Database Connectivity (JDBC) API
- Java Servlet Technology

3.3.1 HARDWARE SPECIFICATION

Cloud Server

A cloud server is a virtual server that runs on a cloud computing environment. It can be accessed remotely from anywhere via the internet. It provides scalable and flexible resources for various applications and services.

Fig 3.3.1 Diagram of Cloud Server



3.3.2 SOFTWARE SPECIFICATION

3.3.2.1 Java Programming Language

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible. It is a general-purpose programming language intended to let application developers write once, run anywhere (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

3.3.2.2 Java Database Connectivity (JDBC) API

JDBC is an API for the Java programming language that defines how a client may access a database. It provides methods for querying and updating data in a database and is oriented towards relational databases.

3.3.2.3 Java Servlet Technology

Java Servlet is a technology which is used to create a web application. It is an API that provides many interfaces and classes including documentation. Servlet is an interface that must be implemented for creating any Servlet.

3.4 INPUT REQUIREMENTS:

1. **Data for Storage:** Input data to be stored securely in the cloud storage mechanism. This may include various types of files, documents, or data objects.

2. **Encryption Parameters:** Parameters required for encryption, such as encryption keys and encryption algorithms. These parameters are necessary to ensure the confidentiality of the stored data.

3. **Data Dispersion Parameters:** Parameters related to data dispersion techniques, such as dispersion algorithms and dispersion factors. These parameters determine how the data is dispersed across multiple storage locations.

4. **Access Control Policies:** Policies specifying the access rights and permissions for different users or user groups accessing the stored data. This ensures proper access control and data privacy.

5. **System Configuration:** Configuration settings for the cloud secure storage mechanism, including storage infrastructure, network settings, and security configurations.

3.5 OUTPUT REQUIREMENTS:

1. **Encrypted Data:** The input data encrypted using the specified encryption parameters. This ensures that the data remains confidential and secure during storage.
2. **Dispersed Data Fragments:** Data fragments generated through data dispersion techniques. These fragments are distributed across multiple storage locations to enhance data security and resilience against attacks.

3. **Access Logs:** Logs detailing access attempts and activities related to the stored data. This includes information about users, timestamps, and accesspermissions.

4. **Storage records:** Detailed records of all actions performed on the stored data, including modifications, deletions, and access changes. Audit trails help ensure data integrity and accountability.

3.6 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposalis put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

1. Economical feasibility
2. Technical feasibility
3. Social feasibility

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.7 DEVELOPMENT ENVIRONMENT

3.7.1 JAVA

It is a Platform Independent. Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

3.7.2 INTRODUCTION TO JAVA

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name *7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The

World Wide Web, a name applied to the Internet's millions of linked HTMLdocuments was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. Thebrowser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

3.7.3 FEATURES OF JAVA

Java is a popular programming language that is widely used for backend development. Here are some of the key features of Java:

1. Platform Independence: Java programs are compiled into bytecode, enabling them to run on any platform with a Java Virtual Machine (JVM).

2. Object-Oriented: Java supports object-oriented programming, promoting modular and reusable code through classes and objects.

3. Rich Standard Library: Java offers a comprehensive standard library (Java API) with classes and methods for tasks like networking, database connectivity, file I/O, and more.

4. Concurrency Support: Java provides built-in support for concurrency through multithreading, allowing developers to create multithreaded applications for improved performance.

5. Security: Java prioritizes security with features such as bytecode verification, runtime security checks, and a robust security manager.

6. Scalability: Java's modular architecture and support for distributed computing frameworks like Java EE and Spring facilitate the development of scalable backend systems.

7. Community and Ecosystem: Java has a large and active community, contributing to a rich ecosystem of libraries, frameworks, tools, and resources.

8. Performance: Java offers high-performance capabilities through optimizations like Just-In-Time (JIT) compilation, runtime profiling, and garbage collection.

3.7.4 APACHE TOMCAT

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Environment

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high- availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

3.7.5 AMAZON WEB SERVICES(AWS)

Amazon Web Services (AWS) is a leading cloud computing platform provided by Amazon.com, offering a comprehensive suite of services to businesses, governments, and individuals worldwide. Launched in 2006, AWS has revolutionized the way organizations build, deploy, and manage their IT infrastructure, providing scalable,reliable, and cost-effective cloud solutions.

At its core, AWS enables users to access computing power, storage, and other IT resources over the internet, eliminating the need for physical hardware and infrastructure management. With AWS, users can quickly provision virtual servers, storage volumes, databases, and more, allowing them to focus on developing applications and services without worrying about the underlying infrastructure.

One of the key features of AWS is its elasticity, which allows users to scale their resources up or down based on demand. Whether a business needs to handle a sudden spike in traffic or reduce costs during periods of low activity, AWS provides the flexibility to adjust resources dynamically, ensuring optimal performance and cost efficiency.

AWS is also known for its reliability, offering redundant infrastructure and high availability features to minimize downtime and ensure business continuity. With data centers (regions) located in multiple geographical locations worldwide, AWS provides global coverage and resilience, enabling users to deploy their applications closer to their customers for lower latency and better performance.

Security is another critical aspect of AWS, with robust features such as data encryption, identity and access management (IAM), network security, and compliance certifications. AWS adheres to industry best practices and compliance standards, ensuring that data and applications are protected against unauthorized access, breaches, and cyber threats.

Furthermore, AWS offers a pay-as-you-go pricing model, allowing users to pay only for the resources they use without any upfront costs or long-term commitments. This cost-effective pricing model, combined with AWS's scalability and flexibility, makes it an attractive choice for businesses of all sizes, from startups to enterprise-level organizations.

Overall, AWS provides a powerful and flexible cloud computing platform that empowers organizations to innovate, scale, and grow their businesses in the digital age, driving agility, efficiency, and competitiveness in today's rapidly evolving market landscape.

3.7.6 FEATURES OF AWS:

- Compute: AWS provides scalable compute capacity through services like Amazon EC2 and AWS Lambda.
- Storage: AWS offers options such as Amazon S3, Amazon EBS, and Amazon Glacier for different storage needs.
- Database: Managed database services include Amazon RDS, Amazon DynamoDB, and Amazon Redshift.
- Networking: AWS provides services like Amazon VPC, AWS Direct Connect, and Amazon Route 53 for networking needs.

- Security and Identity: Features include IAM, AWS KMS, and AWS IAM for robust security.
- Management Tools: AWS offers tools like Amazon CloudWatch, AWS CloudTrail, and AWS Trusted Advisor for managing resources.
- Developer Tools: AWS Code Deploy, AWS Code Commit, and AWS Code Pipeline facilitate development processes.
- AI and Machine Learning: Services like Amazon Sage Maker, Amazon Recognition, and Amazon Polly cater to AI and ML needs.
- Internet of Things (IoT): AWS IoT Core, AWS IoT Greengrass, and AWS IoT Analytics support IoT initiatives.
- Serverless Computing: AWS Lambda, Amazon API Gateway, and AWS Step Functions enable serverless architecture.

CHAPTER 4
SYSTEM DESIGN

4.1 SEQUENCE DIAGRAM

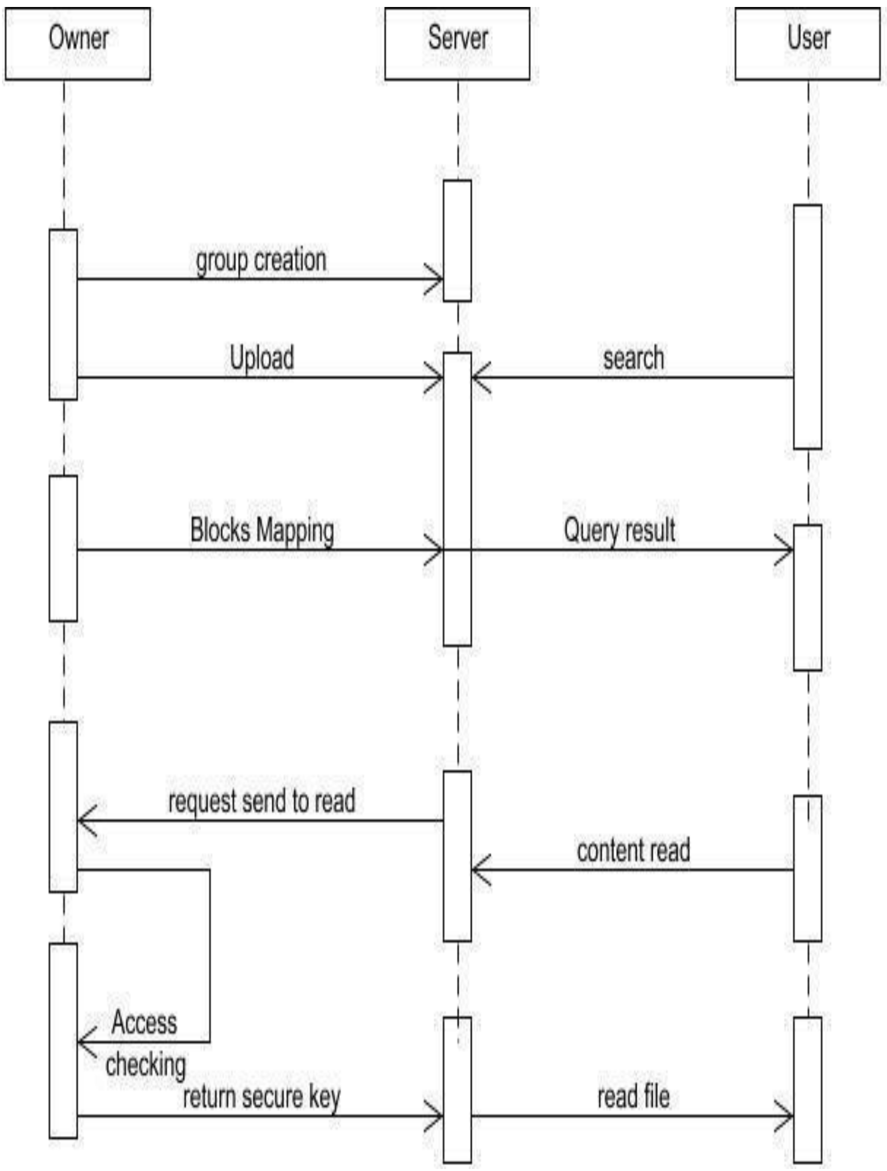
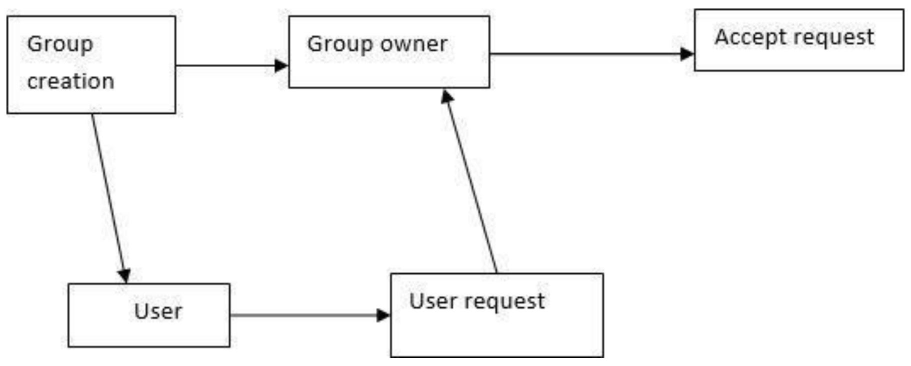


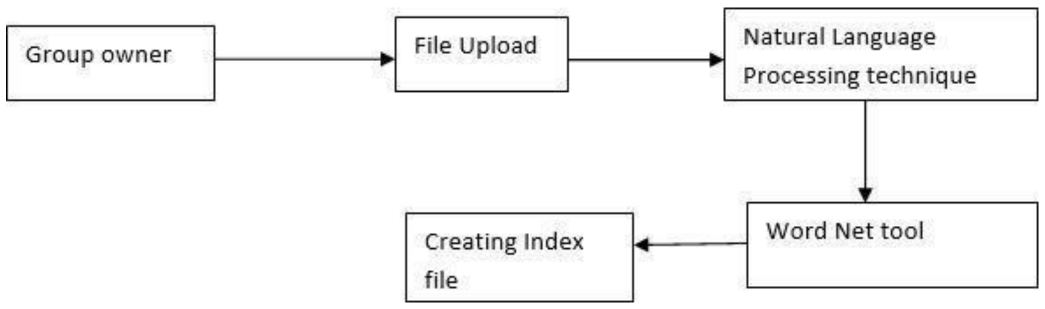
Fig 4.1 Sequence Diagram

4.2 DATA FLOW DIAGRAM

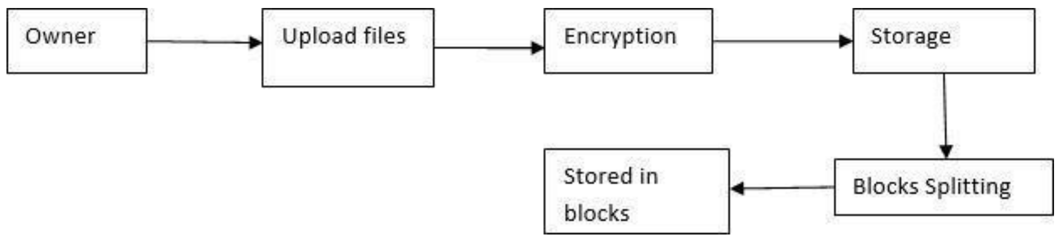
LEVEL 0:



LEVEL 1:



LEVEL 2:



LEVEL 3:

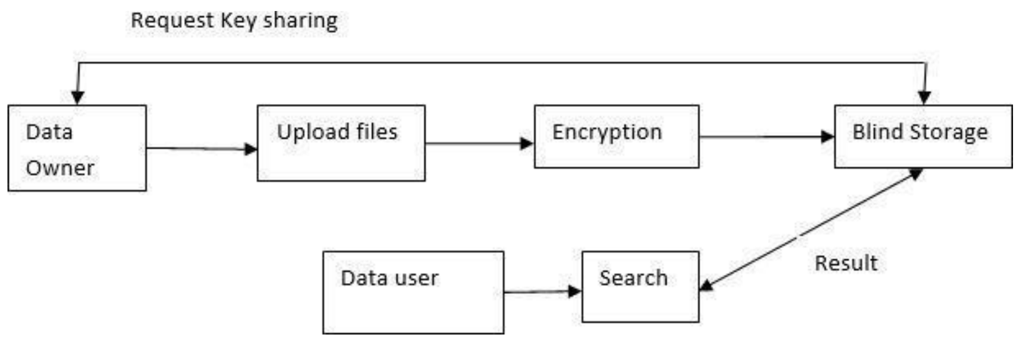


Fig 4.2 Data Flow Diagram

4.3 UML DIAGRAMS

4.3.1 Use case Diagram

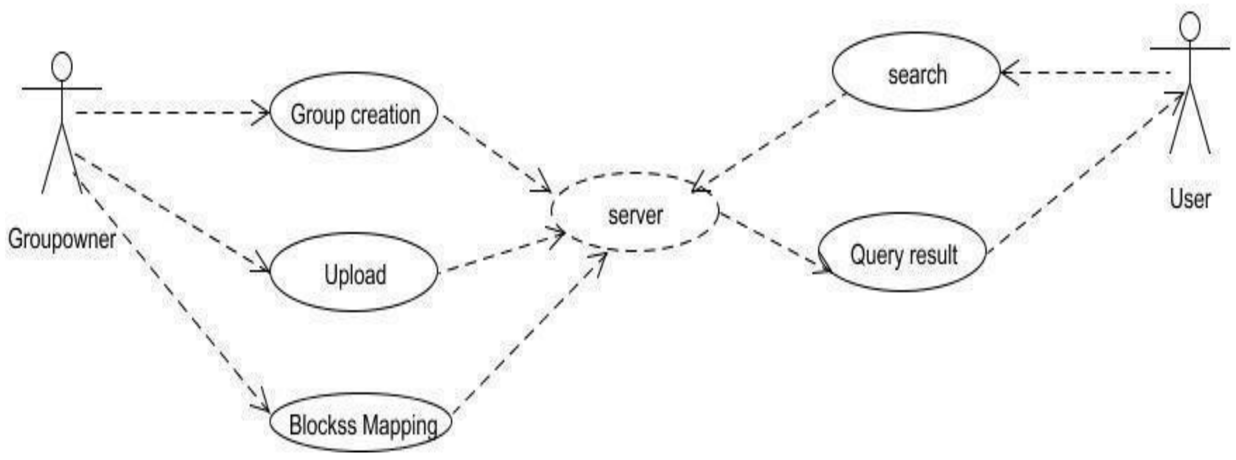


Fig 4.3.1 Use Case Diagram

4.3.2 Class Diagram

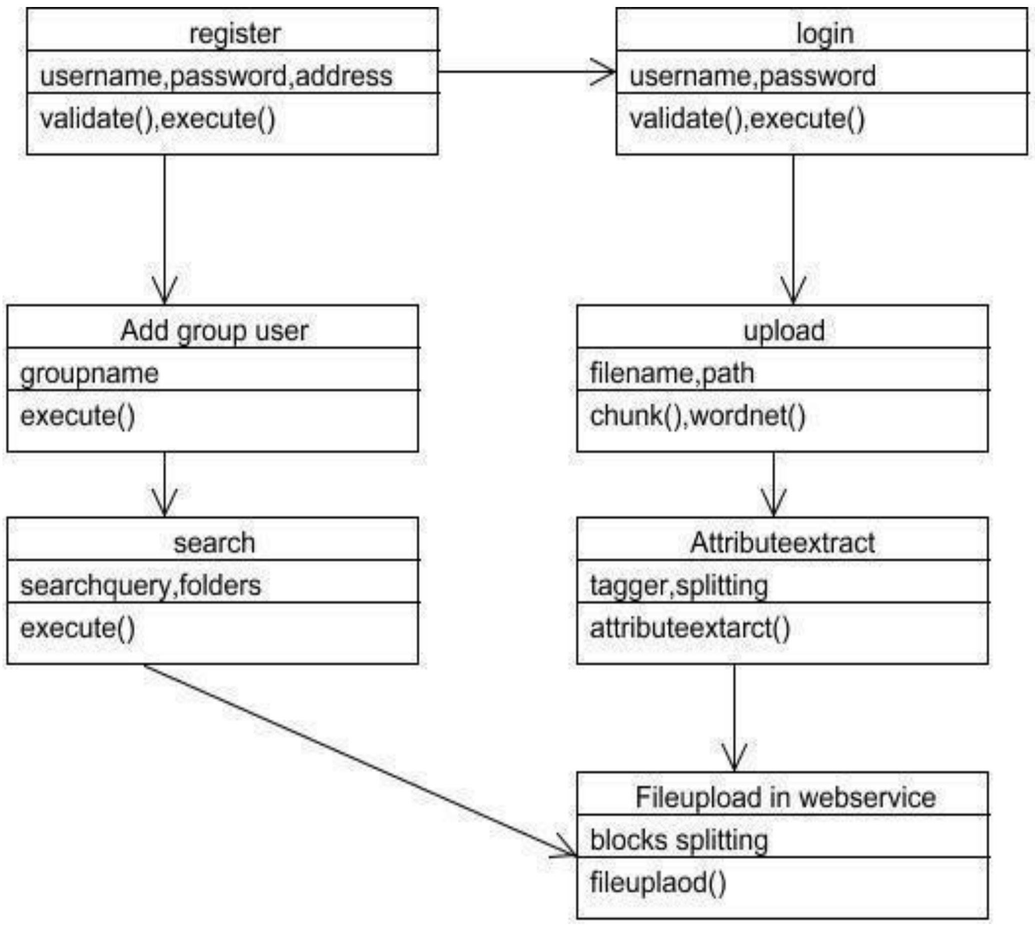


Fig 4.3.2 Class Diagram

4.3.3 Activity Diagram

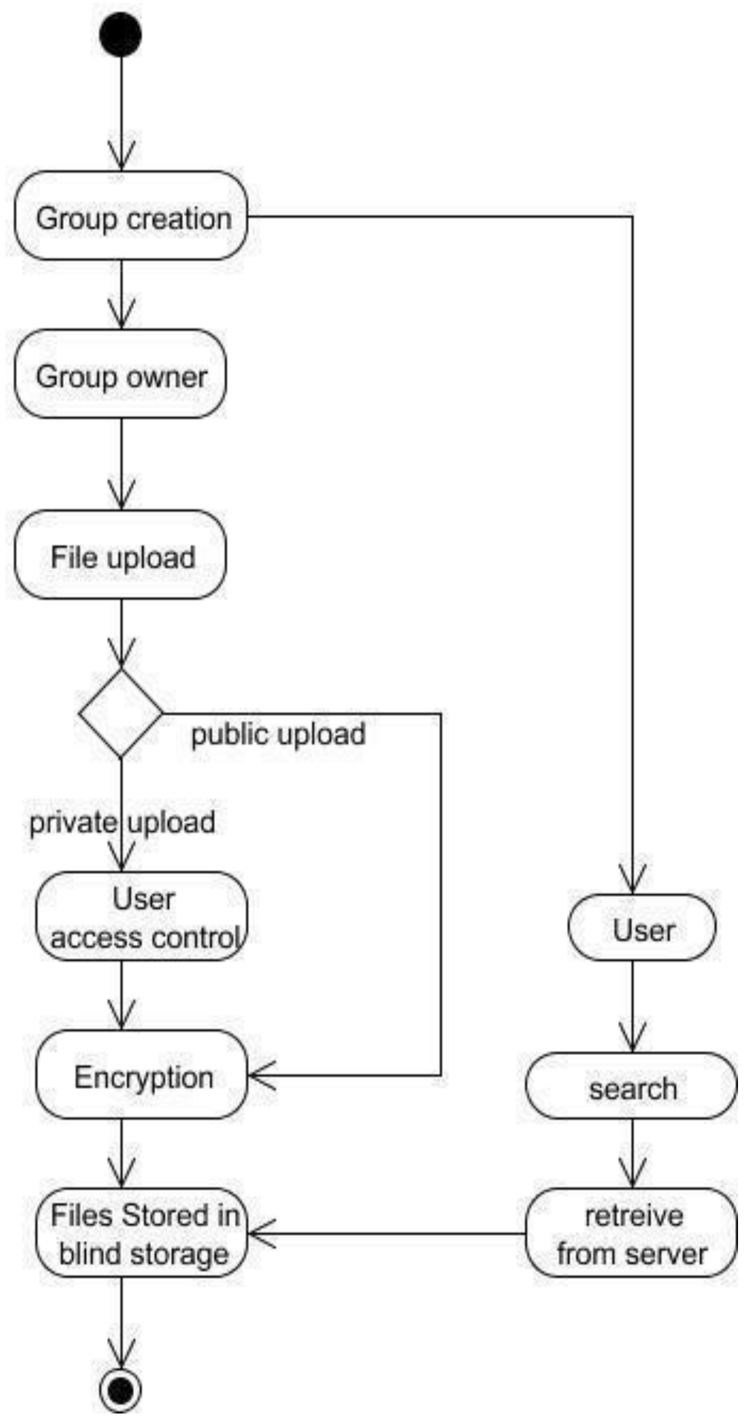


Fig 4.3.3 Activity Diagram

4.3.4 Collaboration Diagram

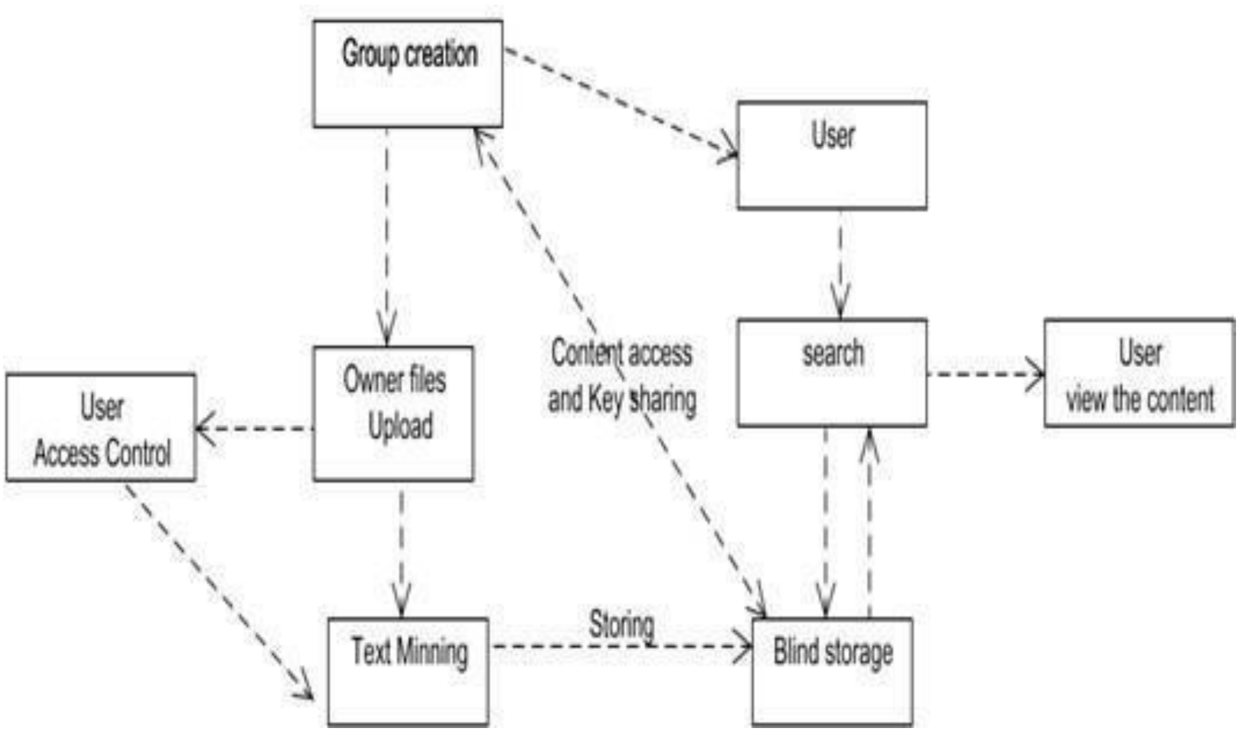


Fig 4.3.4 Collaboration Diagram

CHAPTER 5
SYSTEM ARCHITECTURE

5.1 ARCHITECTURE OVERVIEW

The system architecture delineates two principal roles: the data owner and the data user. The data owner's responsibilities commence with data splitting, wherein the original data undergoes fragmentation to bolster security. Following this, each fragment undergoes encryption using cryptographic techniques to uphold confidentiality. Additionally, a text mining process may be invoked to extract insights from the encrypted data without compromising security. Subsequently, the encrypted data fragments are uploaded to a designated storage location, ensuring access controls and employing secure key sharing mechanisms to regulate access.

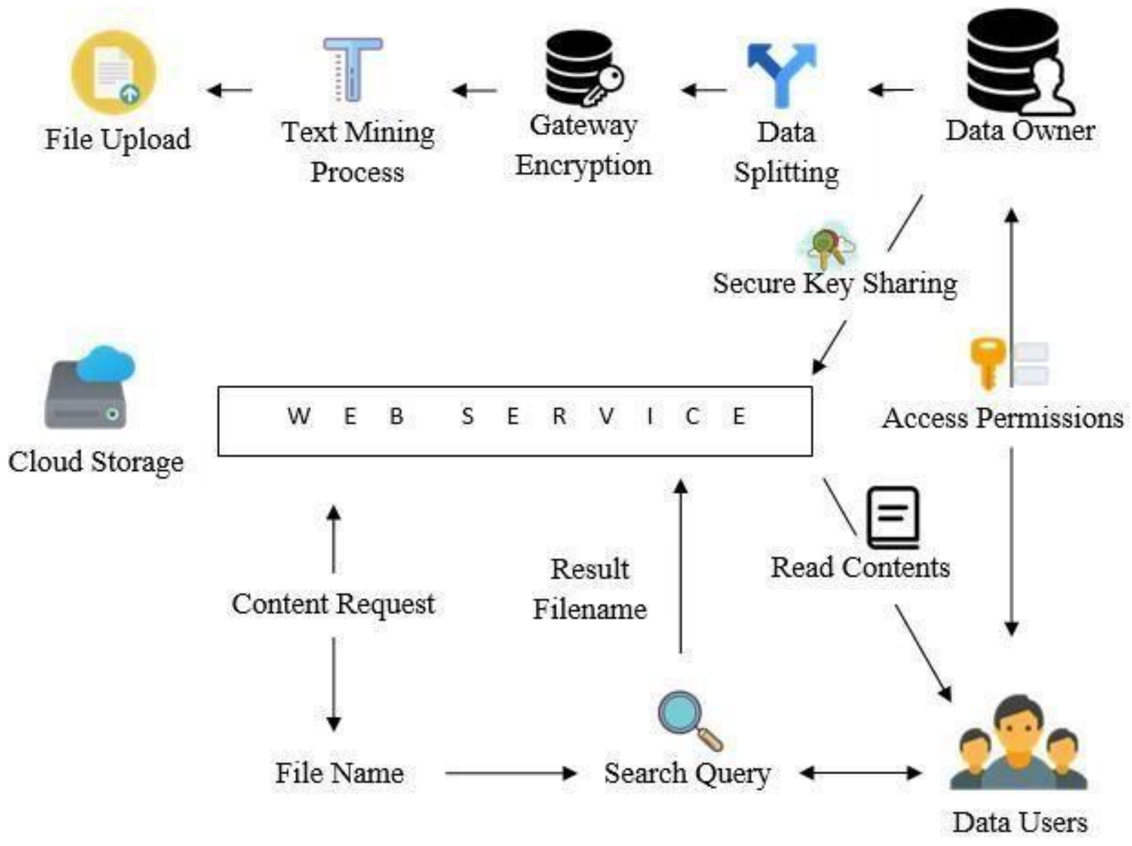


Fig 5.1.1 Architecture Diagram

Conversely, the data user interacts with the system by initiating content requests and search queries. To access specific content, the data user must first obtain authorization from the data owner. Similarly, when searching for specific information, the data user submits a search query to retrieve relevant data fragments.

Upon securing authorization, the data user decrypts and accesses the requested content using encryption keys provided by the data owner. This meticulously structured architecture guarantees the secure storage, retrieval, and access of sensitive data while steadfastly preserving confidentiality and integrity throughout the process.

5.2 MODULE DESCRIPTION

The Cloud Secure Storage Mechanism has four module the 3 modules are related to Input, encryption and user's choice of encryption and Data Dispersion and the last module is about the user accessing the file. The four modules are:

1. Group creation.
2. Text Mining process.
3. Blind storage.
4. Query search.

1. Group creation

In the proposed environment, data owners must first register their accounts, thereby initiating their presence within the system. Upon registration, data owners are empowered to create groups tailored to their specific needs and requirements. Simultaneously, data users are also prompted to register their accounts within the system. To gain access to specific data, data users must request permission from the respective group owner. This request is then forwarded to the data owner, who retains the authority to accept or deny access to the requested data. Notably, the system accommodates the creation of multiple groups, each endowed with an owner responsible for managing access permissions and group dynamics. Within each group, users are granted access to group-owned data exclusively, ensuring the delineation of data access privileges. Importantly, until the data owner formally accepts the user's request, access to the group's data remains inaccessible, thereby maintaining strict adherence to data privacy and security protocols.

2. Text mining process

In this module, the data owner is empowered to upload documents to the system. Through this upload functionality, the data owner can seamlessly transfer files to the system, initiating subsequent processing steps. Upon upload, the content of the uploaded files undergoes extraction using advanced Natural Language Processing (NLP) techniques. Leveraging NLP technology, the system delves into the textual content of the files, extracting keywords pivotal to understanding the underlying context. To enrich this process further, the extracted keywords are subjected to synonymization using the WordNet tool, which provides a comprehensive lexical database of English words and their semantic relationships. The first step in the text mining process involves the implementation of a Part-of-Speech (POS) tagger, facilitating the extraction of keywords from the uploaded files. Subsequently, NLP processes are employed to discern the literal meaning of the extracted keywords, enhancing comprehension and analysis. These keywords are then subjected to analysis within the WordNet API, enabling the system to identify related terms and synonyms, thereby enriching the index file. This index file, generated for each upload by the group owner, serves as a repository of keywords and their associated synonyms, facilitating efficient data retrieval and search operations. Notably, all communication with the cloud server is orchestrated through a web service, ensuring seamless and secure data transfer between the system and the cloud storage infrastructure.

3. Blind Storage

After the Natural Language Processing (NLP) phase, wherein the uploaded data undergoes linguistic analysis to extract meaningful insights, the processed data is routed through a gateway for encryption. This crucial step ensures the confidentiality and integrity of the data before storage. Once encrypted, the data is stored as an index file, consolidating essential information for subsequent retrieval and access. During the data upload process, the owner wields authority over access control and privileges, dictating user permissions and the extent of their rights. Access control delineates whether a user possesses permission to access a file, while privilege delineates the scope of their authority over the data, encompassing read and write permissions. Prior to storage, the file is fragmented into blocks and encrypted utilizing the robust RSA encryption algorithm. These encrypted blocks are then transmitted to the cloud service and stored in blind storage—a secure storage mechanism where documents are segmented into fixed-size blocks. Each block is indexed using a sequence of random integers, and file contents are dispersed randomly across blocks. This randomized distribution ensures that the cloud service can only view encrypted content, safeguarding data confidentiality. Importantly, the encryption key remains solely within the purview of the data owner, ensuring exclusive control over data access and security.

4. Query search

In the system workflow, when a data user initiates a search query in the cloud server, the servers meticulously map the provided keywords and scour the repository for related files. Upon locating relevant files, the cloud server furnishes the user with the corresponding filenames. Should the user wish to access the content of a particular file, they must click on the filename, prompting a request to the cloud server. This request, along with user details and the filename, is forwarded to the data owner for authentication. Leveraging a system where the data owner possesses knowledge of all public keys of users, the data owner encrypts the private key using the public key of the data user. The encrypted key is then transmitted to the server, which subsequently relays this encrypted key to the user. The user decrypts the key using their private key, thus obtaining access to the private key of the data owner. With this cryptographic exchange complete, the data user gains access to the data stored in the blind storage, ensuring secure and authorized data retrieval.

5.3 ALGORITHM USED

- Dynamic Blocks splitting algorithm
- RSA algorithm
- Base 64
- K-means clustering algorithm

5.3.1 Dynamic Blocks splitting algorithm:

Dynamic Blocks Splitting algorithm could dynamically split encrypted data into blocks of variable sizes based on factors such as the size of the data, available storage space, or performance considerations.

Let's define:

D as the total size of the data to be stored.

B as the desired average block size.

N as the number of blocks to be created.

The formula for dynamic block splitting could be based on a heuristic that aims to achieve a balance between storage efficiency and retrieval speed. One possible approach could be to adjust the block size based on the data size and available storage space. Here's a simplified formula:

$$N=[D/B]$$

5.3.2 RSA Algorithm:

The RSA algorithm is a widely used public-key encryption algorithm that enables secure communication over insecure channels.

Encryption:

To encrypt a message M, compute the ciphertext C as $C \equiv M^e (mod n)$.

Decryption:

To decrypt the ciphertext C, compute the plaintext M as $M \equiv C^d (mod n)$

5.3.3 Base64 Algorithm:

Base64 is not an encryption algorithm, but rather an encoding scheme that represents binary data using a set of 64 printable ASCII characters. The purpose of Base64 encoding is to convert binary data into a format that can be safely transmitted over text-based channels, such as email or HTTP.

The Base64 encoding scheme works by dividing the input binary data into groups of 6 bits, which are then represented by a corresponding character from the Base64 character set. The resulting encoded data is typically longer than the original binary data, as each group of 6 bits is represented by one of 64 possible characters.

5.3.4 K-means Clustering Algorithm:

The k-means clustering algorithm is a popular unsupervised learning technique used to partition a given dataset into k clusters. The algorithm aims to minimize the within-cluster variance, which is the sum of squared distances between each data point and the centroid of its assigned cluster.

Here's the basic outline of the k-means algorithm:

Initialization: Randomly choose k data points from the dataset as initial cluster centroids.

Assignment: Assign each data point to the nearest centroid, forming k clusters.

Update: Recalculate the centroids of the clusters based on the mean of the data points assigned to each cluster.

Repeat: Iterate steps 2 and 3 until convergence, typically when the centroids no longer change significantly or a maximum number of iterations is reached.

The formula for calculating the distance between data points and centroids, typically using Euclidean distance, is as follows:

$$\min \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

k is the number of clusters.

C_i is the ith cluster.

μ_i is the centroid (mean) of cluster C_i.

$\|\cdot\|$ represents a distance metric, typically Euclidean distance.

Once distances are calculated for each data point and each centroid, each data point is assigned to the cluster corresponding to the nearest centroid.

The centroid of each cluster is then updated by computing the mean of all data points assigned to that cluster. The mean for each feature dimension is calculated separately.

This process iterates until convergence, as described earlier. The resulting centroids represent the centers of the clusters, and the assignment of data points to clusters forms the final clustering solution.

CHAPTER 6
TESTING

6.1 SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of thedesign itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will beconsiderably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

6.2 TEST DATA AND OUTPUT

6.2.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

6.2.2 FUNCTIONAL TEST

Functional test cases involved exercising the code with nominal inputvalues for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test

- Structure Test

6.2.3 PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

6.2.4 STRESS TEST

Stress Test is the test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

6.2.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

6.2.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another moduleis added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

6.3 TESTING TECHNIQUES / TESTING STRATEGIES

a) TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the wholeset of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise, the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

b) WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity
- Deriving test cases
- Graph matrices Control

c) BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing

d) SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

e) INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

f) PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements ina condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

g) SECURITY TESTING

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

h) VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specifications and are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic.

i) USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

CHAPTER 7
CONCLUSION

7.1 CONCLUSION

Our project addresses cloud data leakage by introducing CSSM, the Cloud Secure Storage Mechanism. CSSM combines data dispersal and encryption to enhance data security and thwart attacks. Experimental results demonstrate CSSM's effectiveness in preventing data leakage with acceptable performance overhead. We also introduce efficient keyword search, ensuring accurate and secure search over encrypted data while preserving privacy through blind storage. Access controls are implemented for each user, enhancing security. CSSM offers a practical solution for cloud storage security, promising robust protection and efficient data management, bolstering trust in cloud computing technologies.

7.2 FUTURE ENHANCEMENT

To address ongoing security challenges, our project plans to enhance our cloud secure storage mechanism. Specifically, we aim to improve key management by implementing dual-server hot backup or cluster approaches for updating the index tree in response to user key changes. Additionally, we propose future enhancements such as creating multiple user groups, enabling multimedia search, adopting asymmetric encryption like RSA, incorporating natural language processing for text mining, and generating index files and keys directly on the cloud. These improvements will bolster our system's resilience against vulnerabilities, reduce the risk of data leakage, and ensurethe confidentiality of sensitive information stored in cloud environments.

APPENDICES

A 1. SDG GOALS:

The project aligns with several Sustainable Development Goals (SDGs) outlined by the United Nations. Here are some relevant SDGs for this project:

1. **SDG 9: Industry, Innovation, and Infrastructure:**
   This project contributes to advancing innovation by developing secure and efficient techniques for outsourcing k-means clustering, thereby promoting technological progress.

2. **SDG 16: Peace, Justice, and Strong Institutions:**
   By focusing on secure data processing using fully homomorphic encryption, the project supports the establishment of strong institutions and the promotion of peaceful and inclusive societies.

3. **SDG 17: Partnerships for the Goals:**
   Collaboration between academia, industry, and governmental institutions is crucial for the success of this project, aligning with the goal of fostering partnerships to achieve sustainable development.

These SDGs highlight the broader societal impact and relevance of the project in promoting innovation, security, and collaboration towards sustainable development.

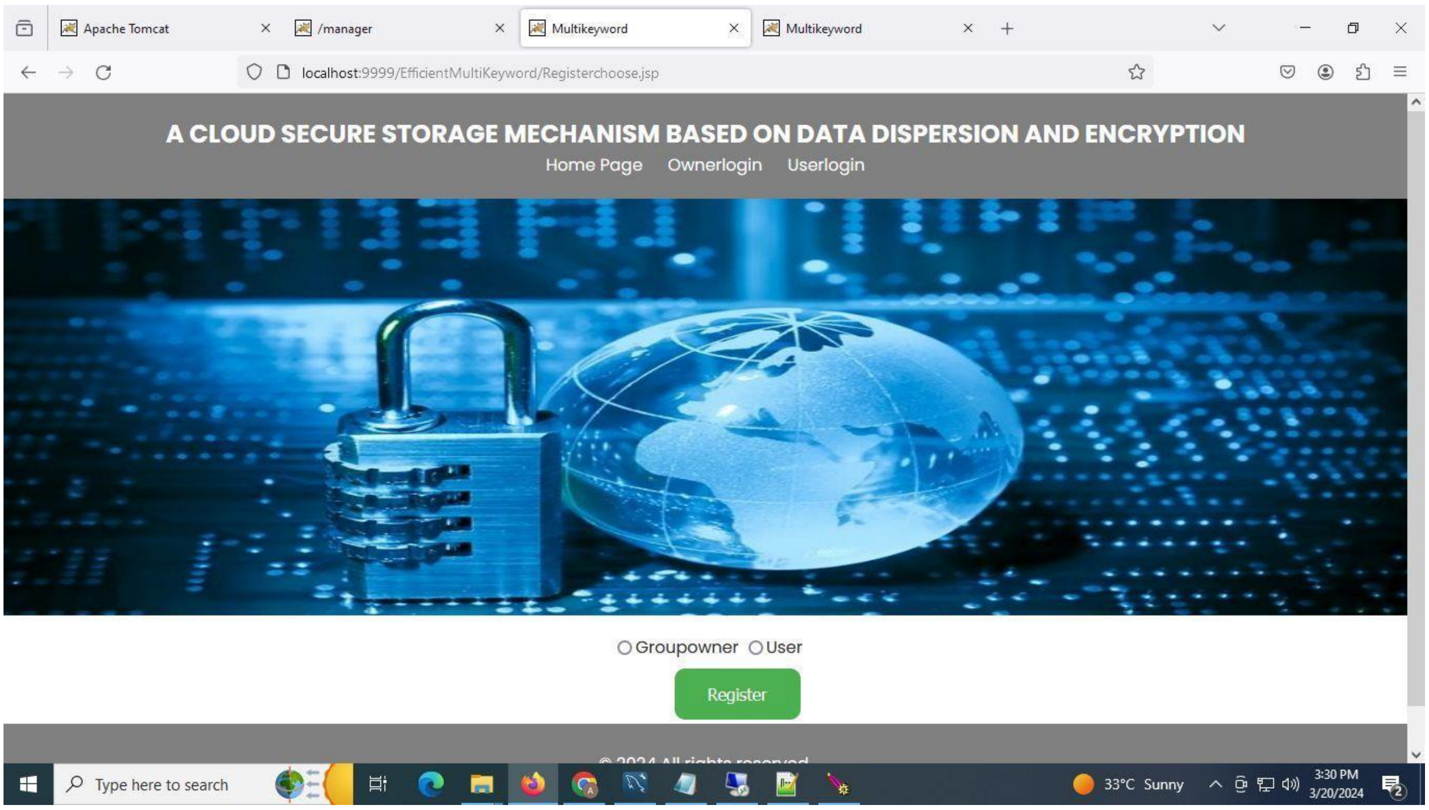
A 2. SOURCE CODE:

```
//File           Upload

package com.logic;

import     java.io.BufferedReader;
import      java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import  java.io.FileReader; import
java.io.FileWriter;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream; import
java.sql.PreparedStatement; import
java.sql.ResultSet;
import java.util.Random;
import java.util.StringTokenizer;
import java.util.Vector;

import com.mysql.jdbc.ResultSetMetaData;

import db.dbservice;

public class FileUpload {

        dbservice dbb=new dbservice();
        Vector mapcon=new Vector();
        public     String    upload(String     encContent,String     serContent,String      fileName,String
userName)
          {

                    String destPath="";
                    String mappingcontent="",grpnam="";
                    PreparedStatement prestmt=null;
                    ResultSet rs=null;
                    String status="";
                    StringBuffer sb=new StringBuffer();
                    String ar[];
                    Vector vec=new Vector();

                    try
                    {
                              File f;

prestmt=(PreparedStatement)dbb.getConnection().prepareStatement(dbb.getgroupname());
                              prestmt.setString(1, userName);
                              rs=prestmt.executeQuery();
                              while(rs.next())
                              {
                                        grpnam=rs.getString("GroupName");

                              }



                              String destPaths = "webapps/ClouddataWebservice/Contents";f=new
                              File(destPaths);
                              if(!f.exists())
                              {
                                        f.mkdir();
                              }
                              destPath = "webapps/ClouddataWebservice/Contents/"+userName;f=new
                              File(destPath);
                              if(!f.exists())
```

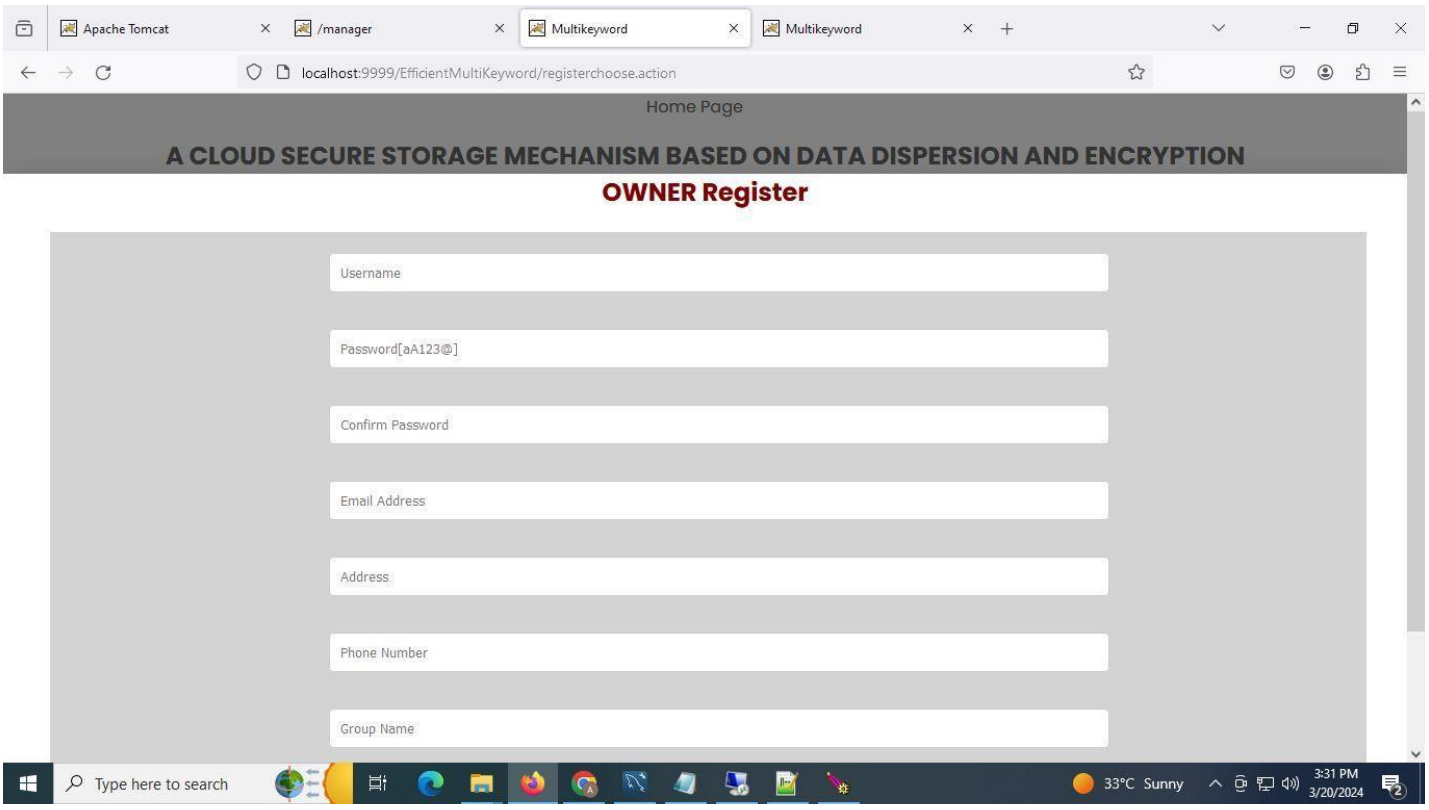
```
                {
                        f.mkdir();
                }
//
//
                StringTokenizer stt=new StringTokenizer(fileName,".");
            String name=stt.nextToken();
            String extenstion=stt.nextToken();
            String newName="";

        newName=name+"Encrypted."+extenstion;
        String aaaa=new String(serContent);
            FileOutputStream                    fileOut                    =new
FileOutputStream("webapps/ClouddataWebservice/Contents/"+"/"+userName+"/"+name+"Index.ser");
          ObjectOutputStream out = new ObjectOutputStream(fileOut);
          out.writeObject(aaaa);
          out.close();
          fileOut.close();

            File blocksfile=new File("webapps/ClouddataWebservice/BLOCKS");
            if(blocksfile.exists()==false)
          {

             new File("webapps/ClouddataWebservice/BLOCKS").mkdir();
            for(int i=0;i<=20;i++){
                    new File("webapps/ClouddataWebservice/BLOCKS/Block"+i).mkdir();
            }
            System.out.println("Cloue Space Created ");
          }
            else{

                    System.out.println("cloud space already created");

            }
```

```
         Random dice = new Random();
       int diceRoll =+dice.nextInt(1000) + 2;

       String pki="PK"+diceRoll; System.out.println("PK=============="+pki);

       /////////////////////////////////////////////////////



       Random dice1 = new Random();
       int diceRoll1 =+dice1.nextInt(1000) + 2;

       String pki1="IP"+diceRoll1;
       System.out.println("PKAAAAAA=============="+pki1);

                                                        File           file1           =           new
File("webapps/ClouddataWebservice/Contents"+"/"+userName+"/"+newName);
       if (!file1.exists())
   {
         String destPath1 = "webapps/ClouddataWebservice/Contents/"+userName;
             System.out.println("newname-------encry==="+newName+"path"+destPath1);
             File ff=new File(destPath1,newName);
                     FileWriter fw = new FileWriter(ff.getAbsolutePath());
                     fw.write(encContent);
                     fw.close();
     int i = 1;// Files count
     System.out.println("file==="+file1);
     InputStream inputStream = new FileInputStream(file1);
         // String videoFile = splitFile1.getAbsolutePath() +"/"+ i +"_"+ file1.getName();//
```

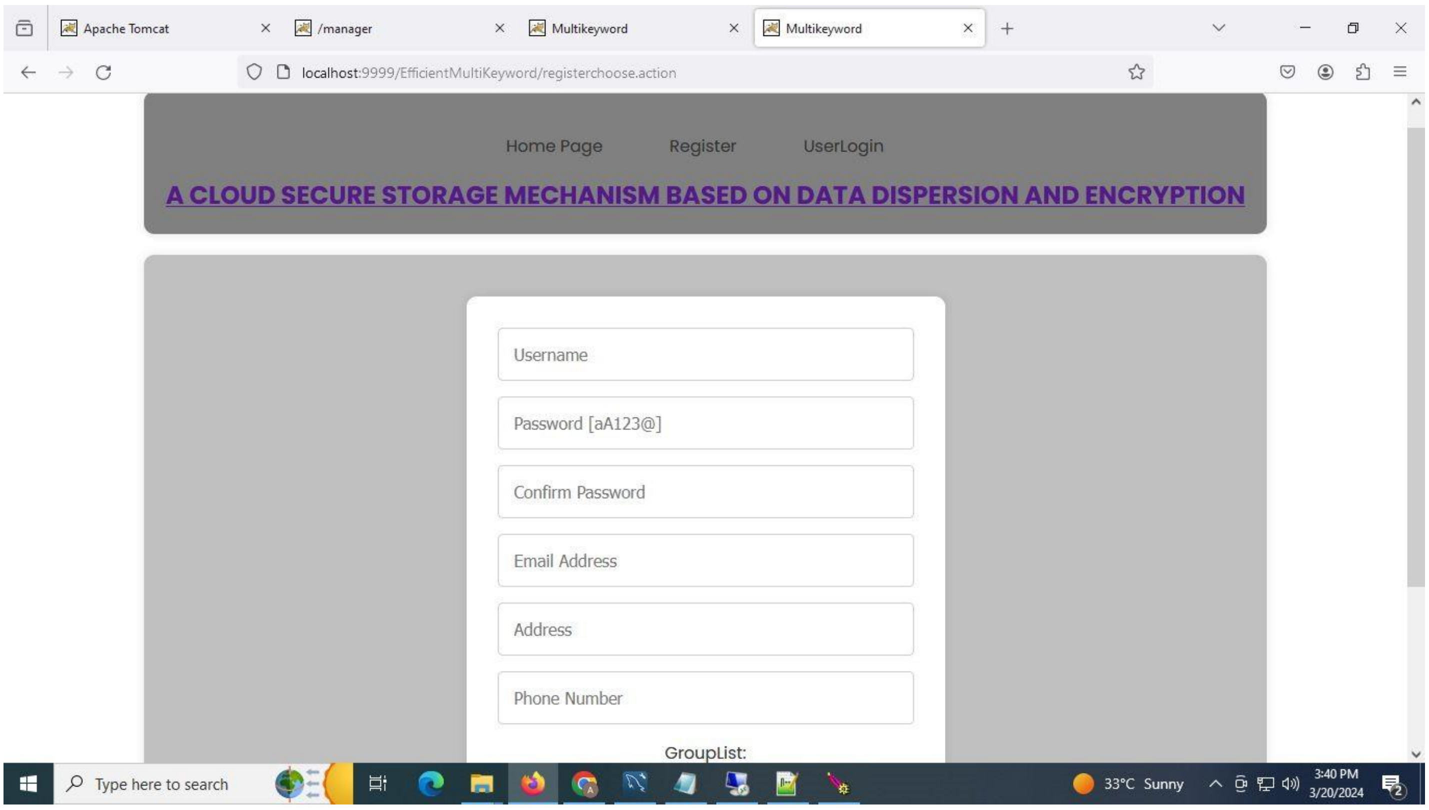
A.3 SCREENSHOTS


Apache Tomcat
/manager
Multikeyword
Multikeyword
localhost:9999/EfficientMultiKeyword/Registerchoose.jsp
A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION
Home Page
Ownerlogin
Userlogin
Groupowner
User
Register
© 2024 All rights reserved
Type here to search
33°C Sunny
3:30 PM
3/20/2024

Register page (user or owner)



Owner registration page



Apache Tomcat | /manager | Multikeyword | Multikeyword

localhost:9999/EfficientMultiKeyword/registerchoose.action

Home Page | Register | UserLogin

A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION

Username

Password [aA123@]

Confirm Password

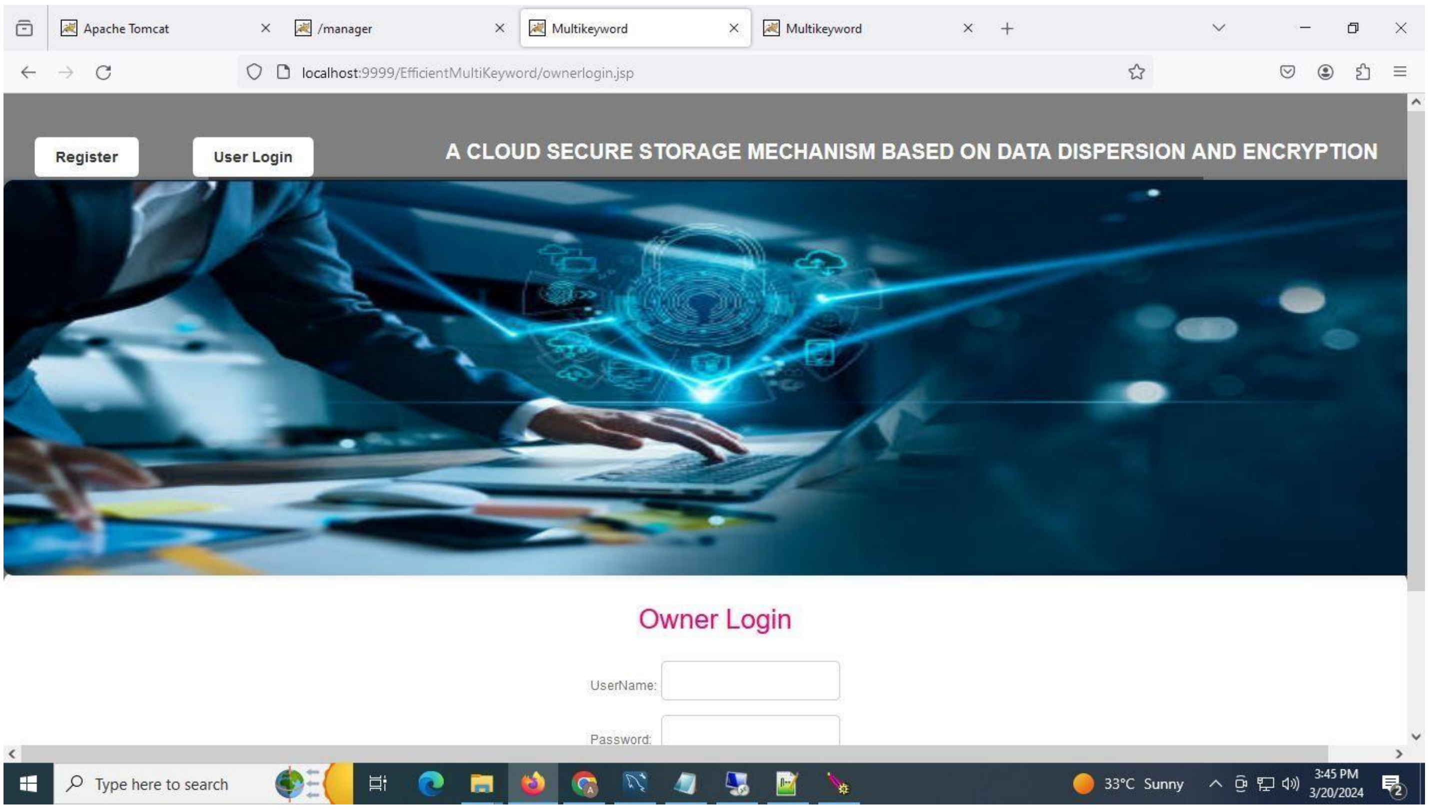
Email Address

Address

Phone Number

GroupList:

User registration page



Apache Tomcat | /manager | Multikeyword | Multikeyword

localhost:9999/EfficientMultiKeyword/ownerlogin.jsp

Register | User Login

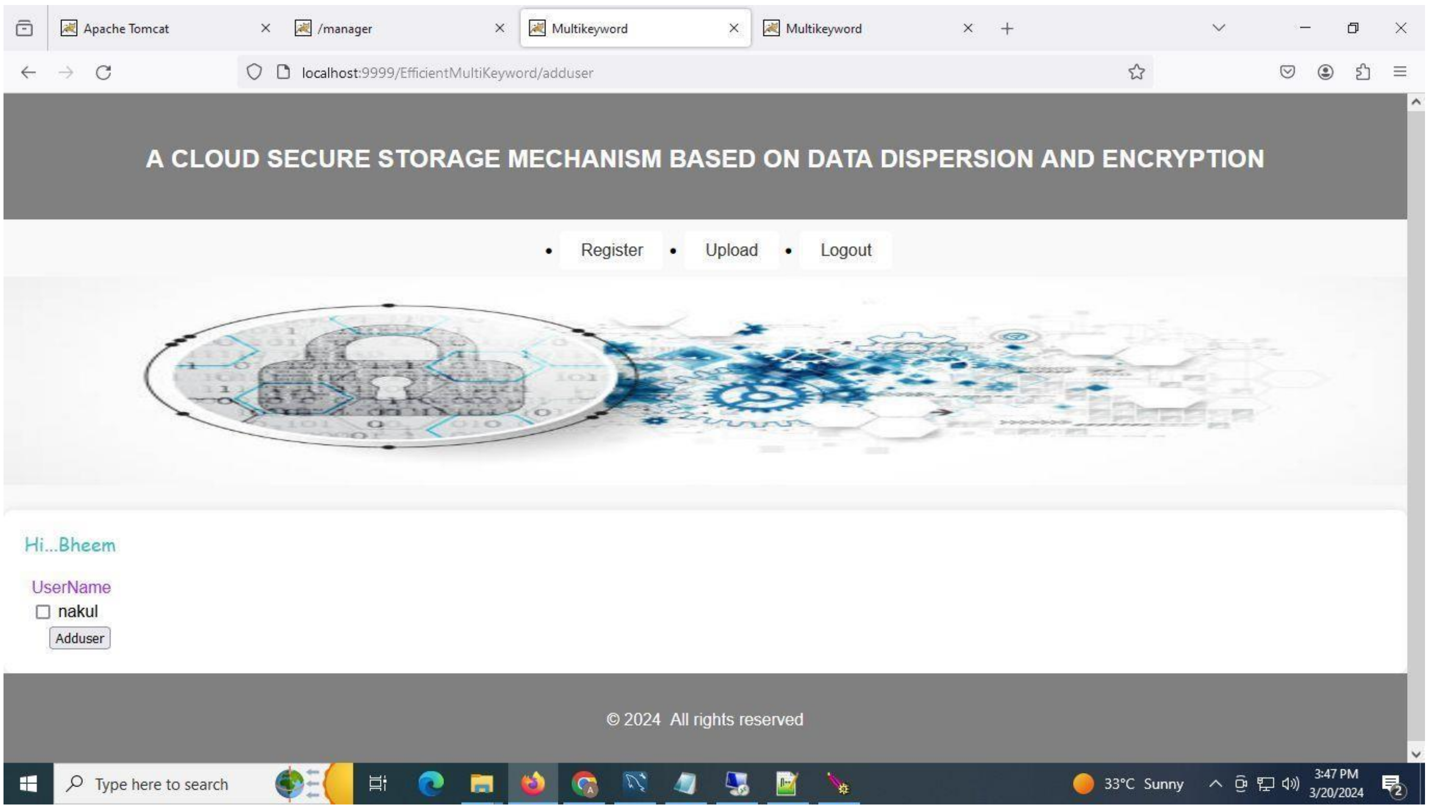
A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION

Owner Login

UserName:

Password:

Owner login page



A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION

Hi...Bheem

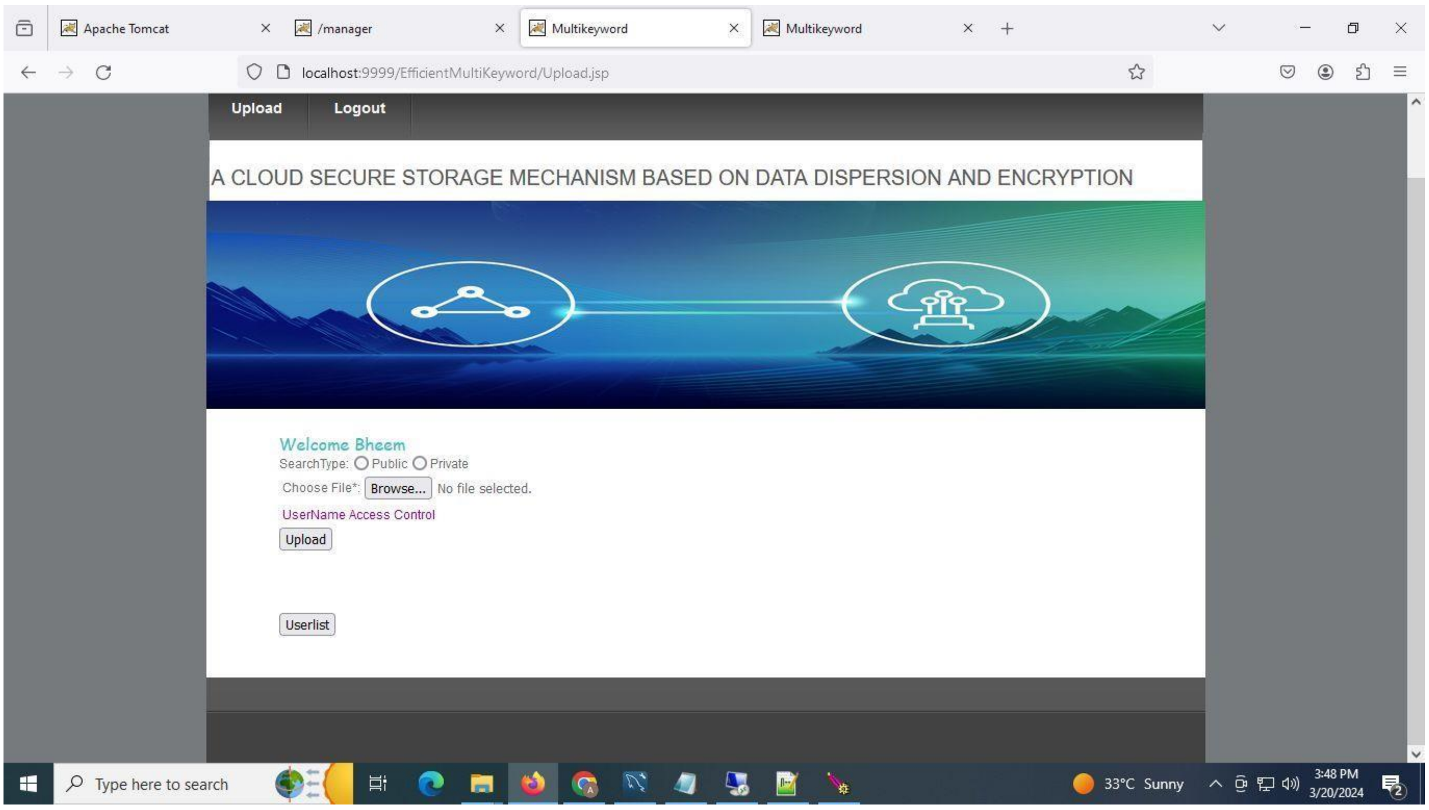
UserName

☐ nakul

Adduser

© 2024 All rights reserved

Adding users to a group done by user



A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION

Welcome Bheem

SearchType: ○ Public ○ Private

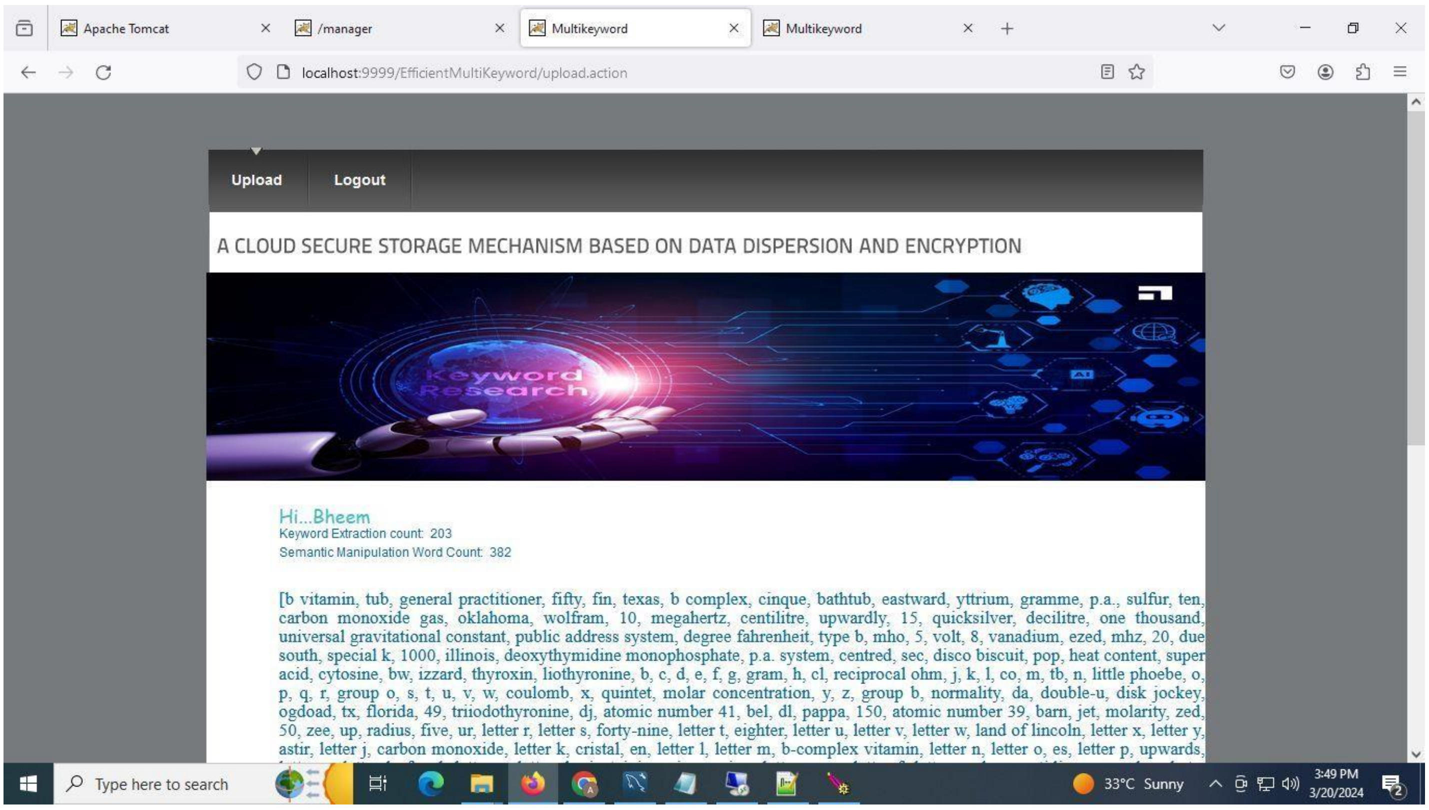
Choose File*: Browse... No file selected.

UserName Access Control

Upload

Userlist

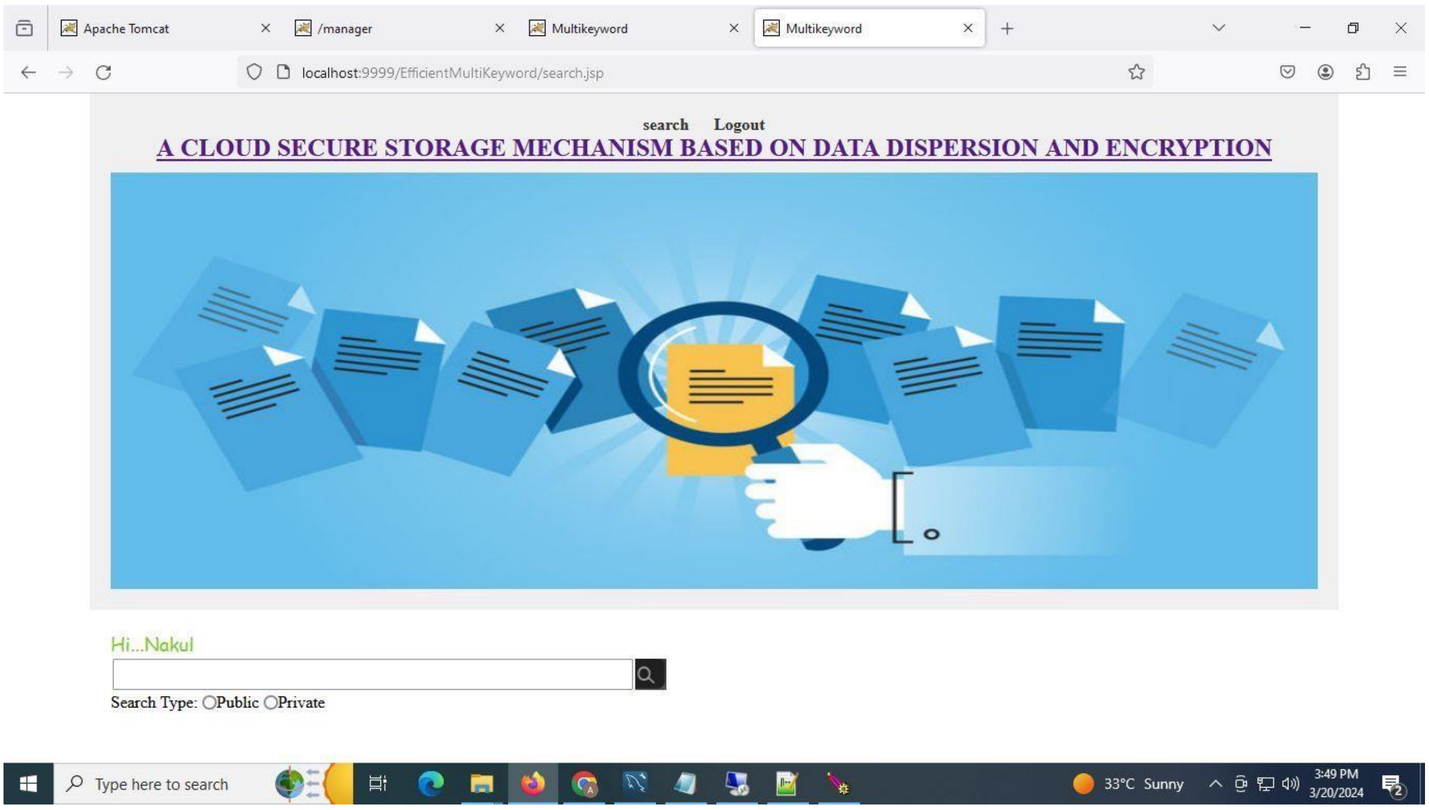
Upload file by owner either in **private**(encrypted using RSA Algorithm)or **public**(encrypted using base64 Algorithm)


Apache Tomcat
/manager
Multikeyword
Multikeyword
localhost:9999/EfficientMultiKeyword/upload.action
Upload
Logout
A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION
Keyword Research
Hi...Bheem

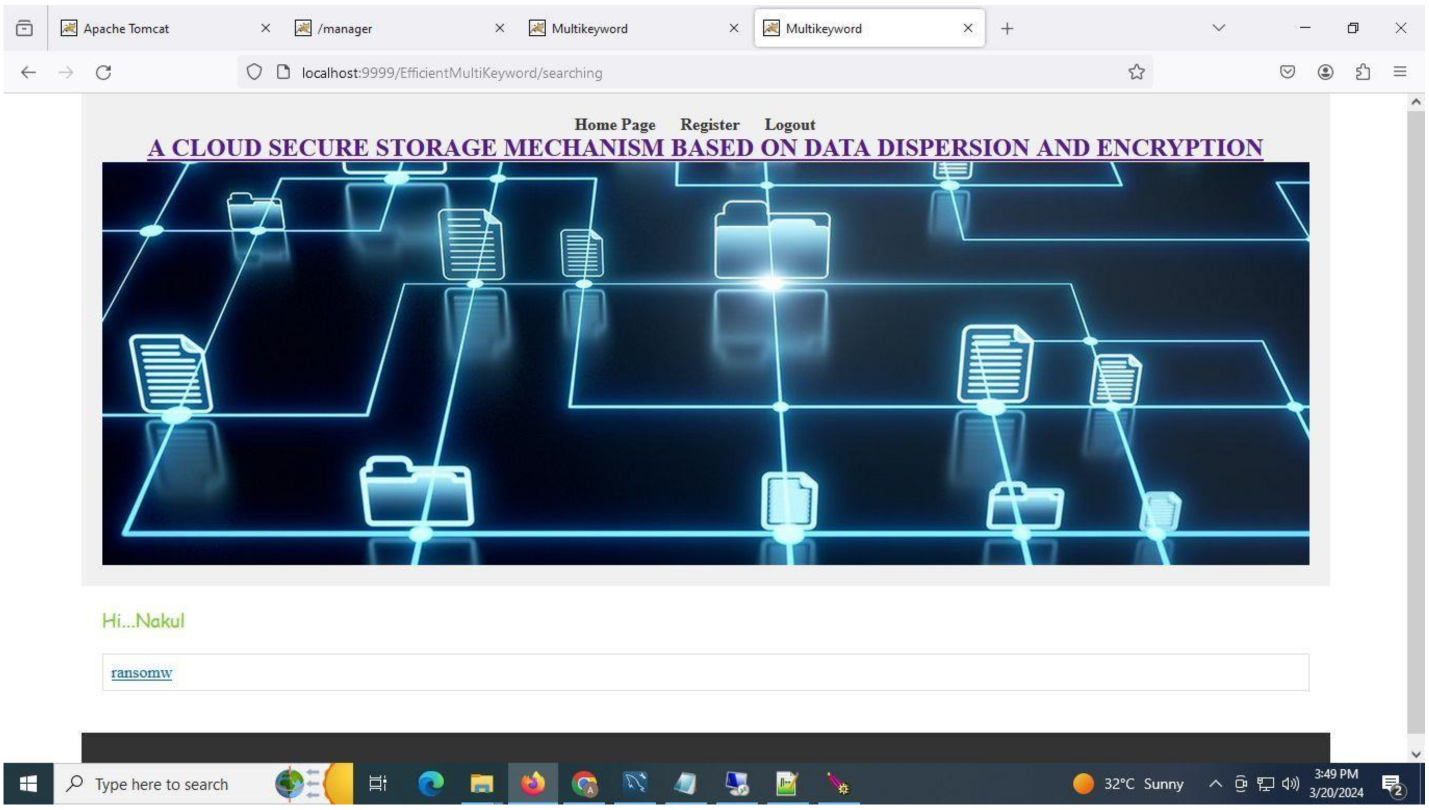
Keyword Extraction count: 203
Semantic Manipulation Word Count: 382

[b vitamin, tub, general practitioner, fifty, fin, texas, b complex, cinque, bathtub, eastward, yttrium, gramme, p.a., sulfur, ten, carbon monoxide gas, oklahoma, wolfram, 10, megahertz, centilitre, upwardly, 15, quicksilver, decilitre, one thousand, universal gravitational constant, public address system, degree fahrenheit, type b, mho, 5, volt, 8, vanadium, ezed, mhz, 20, due south, special k, 1000, illinois, deoxythymidine monophosphate, p.a. system, centred, sec, disco biscuit, pop, heat content, super acid, cytosine, bw, izzard, thyroxin, liothyronine, b, c, d, e, f, g, gram, h, cl, reciprocal ohm, j, k, l, co, m, tb, n, little phoebe, o, p, q, r, group o, s, t, u, v, w, coulomb, x, quintet, molar concentration, y, z, group b, normality, da, double-u, disk jockey, ogdoad, tx, florida, 49, triiodothyronine, dj, atomic number 41, bel, dl, pappa, 150, atomic number 39, barn, jet, molarity, zed, 50, zee, up, radius, five, ur, letter r, letter s, forty-nine, letter t, eighter, letter u, letter v, letter w, land of lincoln, letter x, letter y, astir, letter j, carbon monoxide, letter k, cristal, en, letter l, letter m, b-complex vitamin, letter n, letter o, es, letter p, upwards,

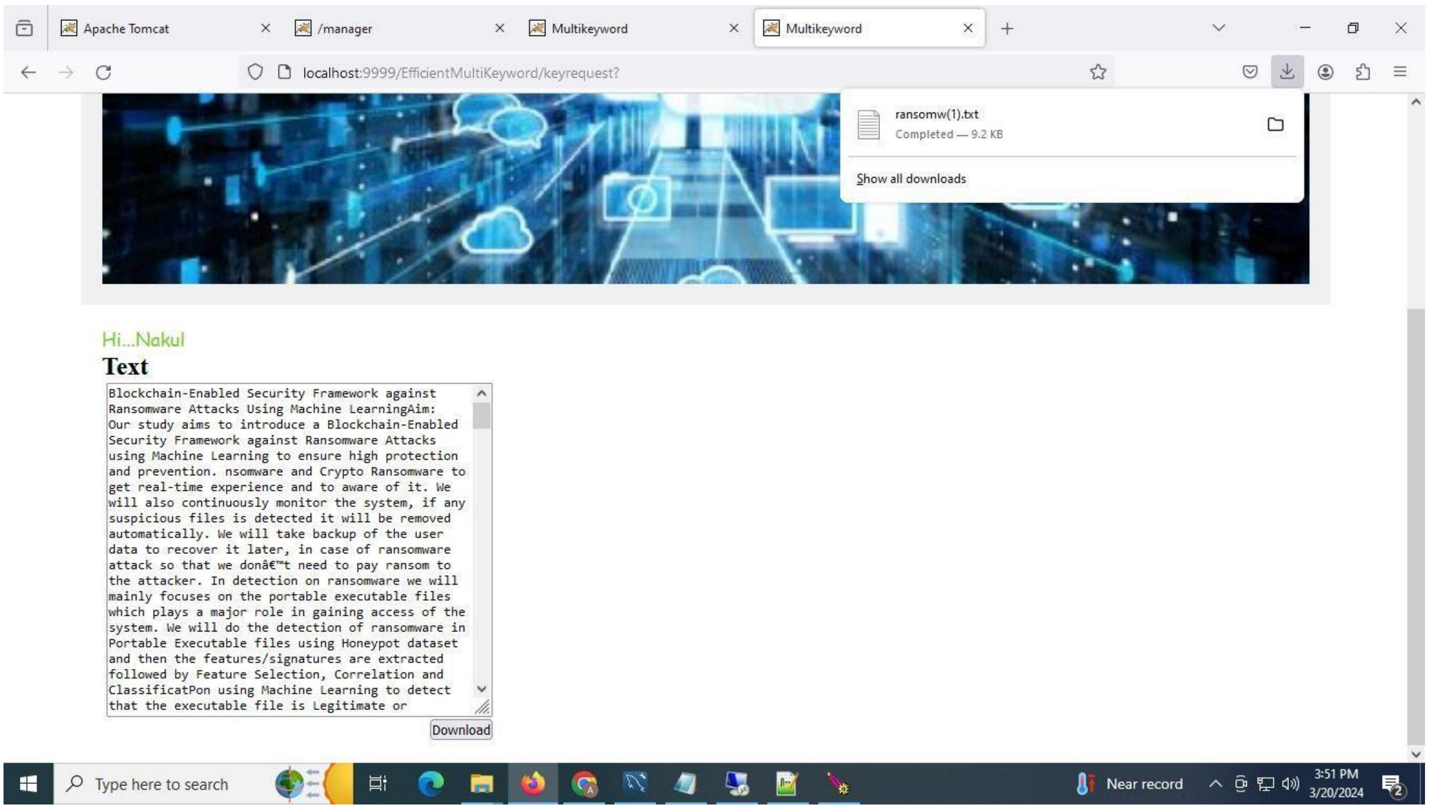
After uploading owners tokenizing the content


Apache Tomcat
/manager
Multikeyword
Multikeyword
localhost:9999/EfficientMultiKeyword/search.jsp
search
Logout
A CLOUD SECURE STORAGE MECHANISM BASED ON DATA DISPERSION AND ENCRYPTION
Hi...Nakul
Search Type: ○Public ○Private
Type here to search
33°C Sunny
3:49 PM
3/20/2024

User page to search for specific private file or public file



File can be viewed after getting access



```
Blockchain-Enabled Security Framework against
Ransomware Attacks Using Machine LearningAim:
Our study aims to introduce a Blockchain-Enabled
Security Framework against Ransomware Attacks
using Machine Learning to ensure high protection
and prevention. nsomware and Crypto Ransomware to
get real-time experience and to aware of it. We
will also continuously monitor the system, if any
suspicious files is detected it will be removed
automatically. We will take backup of the user
data to recover it later, in case of ransomware
attack so that we donâ€™t need to pay ransom to
the attacker. In detection on ransomware we will
mainly focuses on the portable executable files
which plays a major role in gaining access of the
system. We will do the detection of ransomware in
Portable Executable files using Honeypot dataset
and then the features/signatures are extracted
followed by Feature Selection, Correlation and
ClassificatPon using Machine Learning to detect
that the executable file is Legitimate or
```

Access / permission based read or download

REFERENCES

1. Q. Shen, X. Liang, X. Shen, X. Lin, and H. Luo, "Exploiting geodistributedclouds for e-health monitoring system with minimum servicedelay and privacy preservation," IEEE Journal of Biomedical and Health
Informatics, vol. 18, no. 2, pp. 430–439, 2014.

2. H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, "An smdpbased service model for interdomain resource allocation in mobile cloud networks," IEEE Transactions on Vehicular Technology, vol. 61, no. 5, pp. 2222–2232, 2012.

3. Y. Cai, F. R. Yu, and S. Bu, "Cloud computing meets mobile wireless communications in next generation cellular networks," IEEE Network, vol. 28, no. 6, pp. 54–59, 2014.

4. D. Zeng, S. Guo, I. Stojmenovic, and S. Yu, "Stochastic modeling and analysis of opportunistic computing in intermittent mobile cloud," in the 8th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2013, pp. 1902–1907.

5. S. Yu, R. Doss, W. Zhou, and S. Guo, "A general cloud firewall framework with dynamic resource allocation," in Proceedings of ICC, 2013, pp. 1941–1945.

6. F. R. Yu and V. C. M. Leung, "Advances in mobile cloud computing systems," CRC Press, 2015, to appear.

7. Y. Cai, F. R. Yu, and S. Bu, "Dynamic operations of cloud radio access networks (c-ran) for mobile cloud computing systems," IEEE Transactions on Vehicular Technology, 2015, DOI: 10.1109/TDSC. 2015.2406704.

8. K. Zhang, X. Liang, M. Baura, R. Lu, and X. S. Shen, "Phda: A priority based health data aggregation with privacy preservation for cloud assisted wbans," Information Sciences, vol. 284, pp. 130–141, 2014.

9. M. M. Mahmoud and X. Shen, "A cloud-based scheme for protecting source-location privacy against hotspot-locating attack in wireless sensor networks," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 10, pp. 1805–1818, 2012.

10. W. Ren, "uleepp: An ultra-lightweight energy-efficient and privacyprotected scheme for pervasive and mobile wbsn-cloud communications," Ad Hoc and Sensor Wireless Networks, vol. 27, no. 3-4, pp. 173–195, 2015.