Multilevel encryption enabled secure file storage over the cloud

Project submitted to the
SRM University – AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology
In
Computer Science and Engineering
School of Engineering and Sciences

Submitted by

Harshad Dhane (AP19110010341) RKVSS

Mallikarjuna Rao (AP19110010291)

Ashirwad Sharma (AP19110010407)

Under the Guidance of

Dr. Kakumani KC Deepthi

SRM University-AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh - 522 240

[May, 2023]

Certificate

Date: 19-May-23

This is to certify that the work present in this Project entitled “**Multilevel encryption enabled secure file storage over the cloud**” has been carried out by **Harshad Dhane, RKVSS Mallikarjuna Rao, Ashirwad Sharma** under my supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology in **School of Engineering and Sciences**.

Supervisor

____

Dr. Kakumani KC Deepthi

Assistant professor,

Department of Computer Science and Engineering

School of Engineering and Sciences

Acknowledgements

First and foremost, we would like to thank our mentor for this project Dr. Kakumani KC Deepthi for the valuable guidance and advice. She inspired me greatly to work on this project. Her willingness to motivate us has contributed tremendously to our project. We also would like to thank her for her continuous support for the successful compilation of the project. Besides, we would like to thank SRM University-AP for providing us with a good environment and facilities to complete this project. We thank them for offering this opportunity to complete out capstone project. It allowed us to learn and understand the concepts more clearly. Finally, an honorable mention goes to our families, friends for their understanding and support of us in completing this project. Without the help of the people mentioned above, we would have faced many difficulties while doing this.

Table of Contents

Abstract

Cloud computing, also known as distributed computing, emerged before the widespread adoption of large-scale distributed processing technology. These storage systems are commonly employed for tasks such as data backups, scheduled archives, and remote data recovery. The cloud-as-a-service offers an economical and robust solution for storing data over the internet, with a strong emphasis on high availability to guarantee uninterrupted business operations. Security of the stored data is a critical factor influencing continuous data availability in cloud computing. A comprehensive set of procedures and advanced technologies has been developed to identify both external and internal threats to business security. While numerous cloud vendors offer highly advanced and reasonably priced data storage infrastructure, the level of security they provide remains a concern. Hence, the proposed model aims to address the essential security requirements of any cloud data center. The model incorporates security techniques such as AES, DES, and RSA to encrypt file segments, ensuring fast encryption and decryption similar to other symmetric algorithms. The concept of collaboration and consolidation contributes to the principles of information security. A multilevel encryption approach is employed to enhance the security of the cloud server environment, thereby fostering greater trust between cloud providers and their users while ensuring business continuity for organizations. The model addresses the fundamentals of data security and privacy protection, including sensitive data partitioning and access management. It utilizes shared symmetric key cryptography and public key cryptography as cryptographic methods. These techniques use keys to transform data into an unreadable format, providing resistance against most brute force attacks. Furthermore, the model ensures proper authorization of data on the cloud server, aligning with the CIA (Confidentiality, Integrity, Availability) traits. To ensure user-friendly access to data anytime and anywhere, a straightforward frontend interface accompanies the model, enabling continuous data accessibility 24/7.

Statement of Contributions

This is to state that this paper/project developed as part of the final year capstone project at SRM University AP, a team project performed by three individuals namely Harshad Dhane (AP19110010341), RKVSS Mallikarjuna Rao (AP19110010291), Ashirwad Sharma (AP19110010407).

All the team members had equal contribution from the IDEA Discussion phase to the final project execution. This specific project can be divided into 3 parts according to the individual contribution made towards the same, i.e. The frontend and backend development for the project taken care by RKVSS Mallikarjuna Rao (AP19110010291), The cloud connectivity and big data analysis taken care by Ashirwad Sharma (AP19110010407), finally the encryption, decryption and the security part being taken care by Harshad Dhane (AP19110010341).

This project was developed in various stages which is discussed further in this report.

List of Figures

1. Introduction

Distributed computing predates the advent of large-scale processing. According to NIST, the cloud is defined as "a collection of configurable computing resources (systems, storage, applications, services, etc.) that can be rapidly provisioned and released with minimal management effort or service provider interaction." A publication by experts introduced a model aimed at enabling access in a systematic and beneficial manner. The process of registering in the cloud involves more than just recording and programming on the client's computer. With both the application and client application residing on the provider's premises, concerns arise regarding the security of the records. To address this issue, cloud providers can employ cryptographic algorithms to encrypt the documents.

2. Methodology

SDLC (Software Development Life Cycle) - Umbrella Model

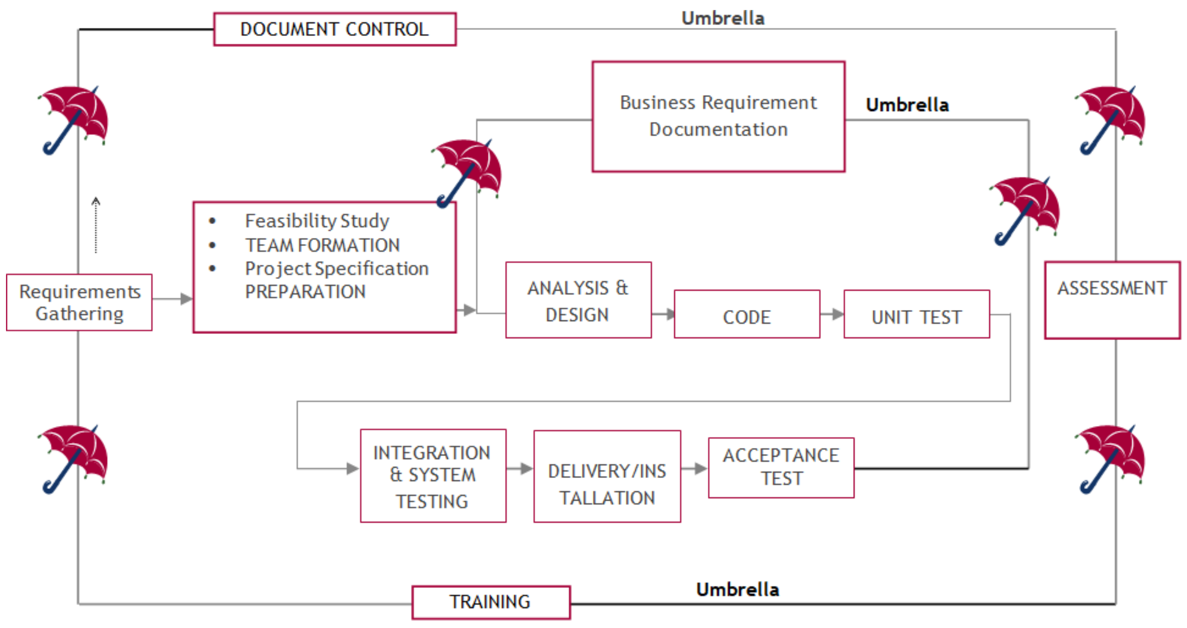


Fig 1: Software development life cycle

The Software Development Life Cycle (SDLC) is a structured framework or process followed to develop high-quality software applications. It consists of a set of sequential phases that guide the development process from the initial concept to the final deployment and maintenance of the software.

2.1 Requirement Gathering Stage

The process of gathering requirements begins by taking the objectives outlined in the high-level requirements section of the project plan as input. Each objective is then carefully examined and transformed into a set of one or more refined requirements. These requirements serve to define the fundamental functionality of the intended application, establish operational and reference data fields, and outline basic data entities. They encompass essential features that encompass critical processes, inputs, outputs, and reports. Furthermore, a user class hierarchy is developed and linked to these core functions, data areas, and data entities. The gathered requirements are documented in a clear and structured manner.

2.2 Analysis Stage

The analysis stage in the Software Development Life Cycle (SDLC) is a critical phase where the requirements gathered in the previous stage are thoroughly examined and translated into a system design. This stage focuses on understanding the functional and non-functional aspects of the software, and it involves activities that lay the foundation for the development process. The analysis helps in determining how the software will be structured and how different modules will interact with each other.

2.3 Designing Stage

The design stage of the Software Development Life Cycle (SDLC) involves creating a detailed plan or blueprint for the software system. It includes activities such as defining the system architecture, specifying internal components and modules, designing the user interface, and creating the database design if applicable. The design stage ensures that the software meets the specified requirements and sets the foundation for its development. It focuses on how the software will be structured, how its components will interact, and how it will fulfill its intended functionality.

2.4 Development Stage

The development stage involves coding, module integration, unit testing, debugging, documentation, version control, and continuous integration (if applicable). It is the phase where the software is developed based on the design specifications, and it includes activities to write and integrate code, test individual components, fix bugs, document the code, manage versions, and ensure continuous integration for seamless collaboration.

2.5 Integration & Test Stage

In the integration and testing phase, the software artifacts, online help, and test data are moved from the development environment to a separate testing environment. This phase involves executing all the test items to ensure that the software is accurate and fully functional. By successfully running the test suite, we can validate the software's migration performance, ensuring its robustness and completeness. Additionally, this phase involves finalizing the reference data for production use and identifying the production users, assigning them suitable roles.

2.6 Literature Survey

This document presents a document security model that provides constructive answers to fundamental security questions in cloud environments. This model uses half-blade encryption where records are scrambled by document fragments and blowfish, and SRNN (modified RSA) for secure communication between client and server.

Information security challenges arise due to the openness and shared ownership characteristics of the cloud environment, rendering traditional security systems inadequate for protecting programs and data stored in the cloud. The problem occurs due to:

• The dynamic nature and flexibility of the distributed computing model in the cloud enable a diverse range of applications and information to be hosted. However, this lack of a specific security framework and limitations poses challenges in terms of identifying and isolating specific assets at risk, as well as making informed decisions when security breaches occur.. Due to the profit transfer model of cloud computing, cloud assets and management can be provided by many suppliers. Given the conflicting circumstances, communicating a collective safety effort is difficult.

• Cloud adoption and multi-tenant sharing of virtual assets may allow customer information to be obtained from other unauthorized customers.

2.7 Multilevel encryption cryptosystem scheme

A hybrid encryption system is used with the goal of ensuring the security of records in the cloud. The record is encrypted by the server and the final hashed document is stored on the server side, since the remote server is assumed to be trusted. Heterogeneous encryption systems refer to the use of multiple encryption algorithms or techniques within a single system or environment. Instead of relying on a single encryption method, heterogeneous encryption systems combine different encryption algorithms to enhance security and provide a higher level of protection for sensitive data. This approach allows for increased complexity and diversity in encryption, making it more challenging for attackers to decipher the encrypted information. By leveraging the strengths of various encryption techniques, heterogeneous encryption systems offer robust and comprehensive data security.

3. System Analysis

The Systems Development Life Cycle (SDLC) is a structured approach used in software development to guide the entire process from initial planning to implementation and maintenance. It consists of a series of phases that encompass activities such as requirements gathering, analysis, design, development, testing, deployment, and maintenance. The SDLC provides a framework for efficiently and effectively developing high-quality software systems, ensuring that they meet the specified requirements, are reliable, and align with the needs of the users and stakeholders.

3.1 EXISTING SYSTEM:

The current system in place is a manual and time-consuming process that lacks automation. There is a need for automating the existing system to streamline operations, improve efficiency, and save time.

3.2 PROPOSED SYSTEM:

To ensure utmost security, your saved image files undergo encryption using a triple-layered approach. AES, DES, and RC6 encryption techniques are employed, providing robust protection for your files and ensuring their complete security.

Advantages:

- Since the key is embedded with the LSB in the image, the key is also secure.
- This approach ensures that the files are protected by a diverse set of encryption techniques. It minimizes the risk of any single encryption algorithm being compromised or exploited.

- The data is securely stored in cloud servers, ensuring that unauthorized access is effectively prevented.
- If one encryption algorithm becomes outdated or vulnerable, the other two algorithms can still provide protection. This future-proofs the encrypted files against potential security threats.

Disadvantages:

- To establish a connection with the cloud server, it is necessary to have an active internet connection at all times.

Application:

Ensuring data security is of utmost importance, particularly when dealing with sensitive information. This system provides a robust solution that can be effectively implemented in sectors such as banking and corporate environments. By employing this system, the secure transfer of confidential data is facilitated, reducing the risk of unauthorized access, data breaches, or information leakage.

4. Implementation

4.1 Introduction of Technologies Used

Java is a versatile programming language used for a wide range of applications. It offers platform independence, enabling programs to run on different operating systems. Java is popular for developing desktop, web, mobile, and enterprise applications, and it is the primary language for Android app development. Java's extensive libraries, scalability, and multi-threading capabilities make it suitable for scientific and financial applications. Its versatility, portability, and large developer community contribute to its widespread adoption in various domains.

4.2 Applications and applets

An application refers to a standalone software program that performs specific tasks or provides certain functionalities for end-users. Applications are typically designed to run on a computer or mobile device, and they can range from simple tools to complex software systems. On the other hand, Java applets are small applications written in the Java programming language that can be embedded and executed within a web browser or applet viewer. Applets are designed to provide interactive features and dynamic content on webpages. They were popular in the early days of the internet for creating interactive graphics, animations, and user interfaces.

4.3 Java Architecture

Java architecture refers to the structure and components that make up the Java programming language and its runtime environment. It includes the Java Development Kit (JDK) for application development, the Java Virtual Machine (JVM) for executing Java bytecode, and the Java Class Libraries (JCL) that provide pre-built functionality. The Java Runtime Environment (JRE) allows users to run Java applications without the full development capabilities of the JDK. Java also offers a wide range of APIs for different functionalities and provides development tools like IDEs for easier coding.

Some given features of JAVA are:

1. **Simple:** Java has a straightforward and easy-to-understand syntax that resembles popular programming languages like C++. Its syntax is designed to be readable and concise, making it easier for developers to write and understand code. It eliminates certain complexities found in C++, such as explicit memory management, pointers, and operator overloading. This is why Java has a reputation for being a more beginner-friendly and accessible language compared to C++.

2. **Object-oriented**: Java is based on the principles of object-oriented programming (OOP), which promotes modular and reusable code. OOP concepts like classes, objects, and inheritance make it easier to organize and structure code, resulting in more maintainable and scalable applications.

3. **Robust:** Java's "write once, run anywhere" principle allows developers to write code on one platform and run it on any platform with a Java Virtual Machine (JVM). This platform independence simplifies the development process and makes it easier to deploy Java applications on different operating systems. Java incorporates automatic memory management through garbage collection. Developers do not have to worry about manual memory allocation and deallocation, reducing the chances of memory leaks and other memory-related issues.

5. Software Requirement Specification

5.1 Requirement Specification

The software requirement specification ensures the provision of a secure storage solution for web servers. It encompasses the necessary software and hardware resources that need to be installed on the server to ensure optimal performance for the application. These requirements align with the standard operating system specifications and ensure compatibility for seamless application development.

5.1.1 Hardware Requirements:

The hardware requirements specify each software interface and system hardware elements. These hardware requirements include configuration features.

- System : Can operate on Pentium processors
- Hard disk capacity: 10 GB

5.1.2 Software Requirement

Software requirements define the utilization of necessary software products, including data management systems, by specifying their required number and version. Each software target interface outlines the specific interface associated with the respective software product.

- OS: Windows
- Coding language: HTML, Java Script, JAVA (JSP Servlet)
- Database: MySQL
- IDE: Netbeans
- Server: Tomcat 7.0

5.2 Functional Requirements:

Functional requirements pertain to the specific system requirements in a highly computerized and code engineering manner. These requirements focus on the desired behavior of the software package and the implementation of business process technology. The primary objective of capturing these functional requirements is to outline the practical aspects of the software's behavior and its alignment with the high-level product style and implementation.

5.3 Non-Functional Requirements:

Non-functional requirements encompass all other requirements that are not included in the aforementioned specifications. For example, a system might need to display the number of records to the user during information presentation. If real-time updates are required, the system designer must ensure that the displayed record count can be dynamically changed within an acceptable timeframe that aligns with the dynamic nature of the record volume.

5.4 Performance Requirements:

Performance is evaluated based on the output delivered by the application. The requirements specification plays a vital role in system analysis, as it enables the design of a system that aligns with the intended environment. Since the users of the existing system will ultimately utilize the new system, it is primarily their responsibility to provide the necessary specifications. Obtaining the requirements early in the process allows for appropriate system design, as modifying a system after its design is challenging, and designing a system that does not meet user needs is futile.

The requirement specification for any system typically includes the following aspects:

1. **Interface with the existing system:** The new system should have the capability to seamlessly integrate and communicate with the current system in use. This ensures smooth data flow and compatibility between the systems.

2. **Accuracy:** The new system should be highly accurate in its operations, ensuring precise and reliable results. This requirement emphasizes the need for correct calculations, data processing, and overall system performance.

3. **Improvement over the existing system:** The new system should offer enhancements and improvements compared to the current system. It should address the limitations or shortcomings of the existing system, providing additional features, efficiency, or effectiveness.

It is important to note that the existing system heavily relies on the user to carry out all tasks and operations. The requirement for the new system is to alleviate this dependency by automating processes, streamlining workflows, and reducing the manual effort required. This shift towards a more automated and user-friendly system aims to improve productivity and reduce human errors.

6. System Design

6.1 System Architecture

The design phase is responsible for making decisions and creating the necessary documentation. It marks the transition from the subject area to the response area. During this phase, system requirements are fulfilled, and the outcome is a document that serves as a blueprint for subsequent stages like implementation, testing, and maintenance.

The design phase is typically divided into two distinct phases: system design and detailed design. The system design, also known as top-ranking style, focuses on identifying the necessary modules, defining their specifications, and establishing how they collaborate to achieve specific outcomes. On the other hand, detailed design, also known as prudential style, concentrates on programming the logic of all modules, ensuring their cohesive functionality.

Overall, the design phase is crucial for shaping the structure and behavior of the system, providing a roadmap for the development team to follow and ensuring the system meets the desired objectives.

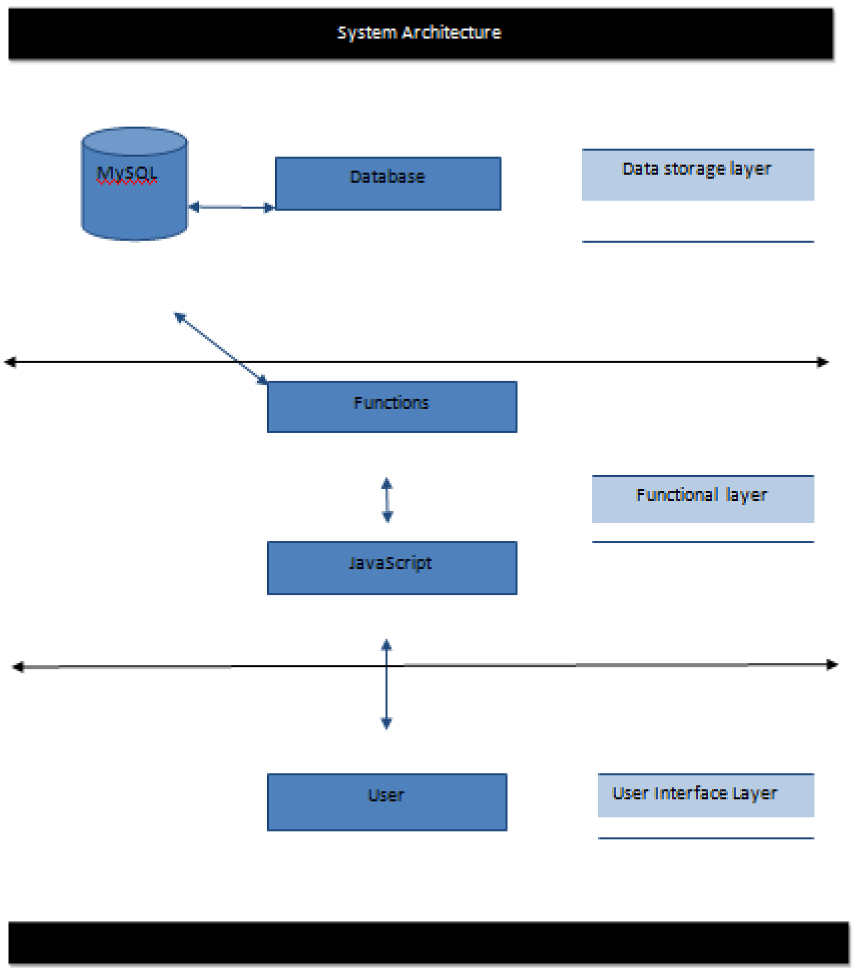


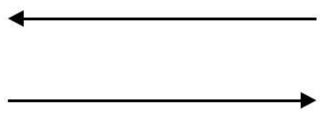
Fig 2: System Architecture

6.2 DFDs (Data flow Diagrams)

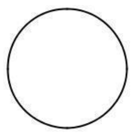
A Data Flow Diagram (DFD) is a visual representation of how data moves within a system. It shows processes, data inputs and outputs, and data stores. DFDs help understand the system's structure, identify data flow and transformations, and communicate system requirements. They aid in analyzing and improving system efficiency and are valuable for system designers, developers, and users to understand data movement and make informed decisions.

The main symbols used to create DFDs are:

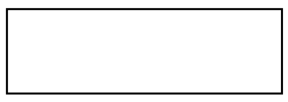
Dataflow:



- **Process:**

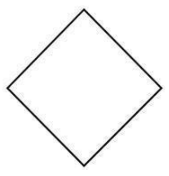


- **Source:**



- **Data Store:**

- **Rhombus**: decision



6.3 UML Diagrams

UML diagrams are visual representations used in Unified Modeling Language (UML) to depict different aspects of a software system. They provide a standardized way to communicate and document system designs, structures, behaviors, and interactions. UML diagrams serve as a common language for software engineers, designers, and stakeholders to understand, analyze, and communicate system requirements and designs. Each view is defined by a series of graphs, such as:

User Model View

This shows the system view from the user's point of view. Analytical statements describe usage scenarios from the end user's perspective.

- Structural model view

  The structural model view focuses on the static structure and organization of the system. It represents the components, relationships, and interactions between different elements of the system. This view includes diagrams like class diagrams, component diagrams, and deployment diagrams, which illustrate the system's architecture, modules, classes, interfaces, and their relationships.

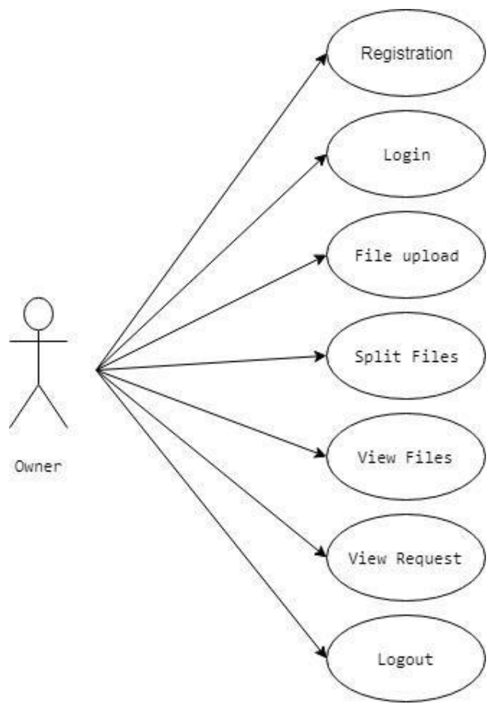
- Behavior model view

  The behavioral model view focuses on the dynamic behavior and functionality of the system. It describes how the system responds and interacts with users, external entities, and other system components. The behavioral model view helps in understanding how the system behaves and how different components interact to achieve specific functionalities.

- Implementation model view

  The implementation model view focuses on the technical aspects of implementing the system. It includes detailed information about the software architecture, design patterns, frameworks, technologies, and coding practices used in the system's implementation.

6.3.1 Use case diagram



A commonly used and straightforward diagram is the one illustrating the interaction between users and the system, highlighting their specific actions and requirements. These diagrams can represent various user types and their distinct interactions with the system. Typically, these diagrams are accompanied by textual descriptions and are often combined with other types of diagrams to provide a comprehensive representation of the system's functionality.

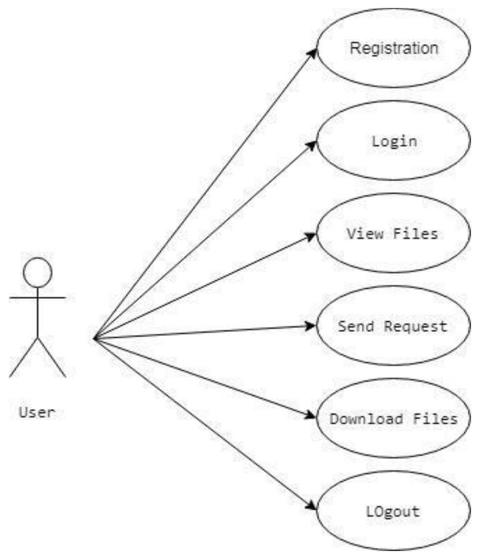


Fig 3 : User and Owner UML Diagram

6.3.2 Class Diagram

Class diagrams are a type of structural diagram in object-oriented modeling that depict the structure and relationships of classes within a system. They provide a visual representation of the static elements of a system, such as classes, attributes, methods, and their associations. Class diagrams help in understanding the overall architecture of a system, the relationships between different classes, and how they collaborate to achieve the system's functionality. They serve as a blueprint for developers, providing a visual guide for designing and implementing the system.

A class shown with a box containing three parts:

1. **Class Name:** This is the name of the class and is typically written in the center or top portion of the box. It represents a blueprint or template for creating objects of that class.
2. **Attributes:** These are the data variables or properties associated with the class. They represent the state or characteristics of objects created from the class. Attributes are listed below the class name and typically include the name and type of the attribute.
3. **Methods:** These are the functions or operations that can be performed by objects of the class. Methods define the behavior or actions that objects can exhibit. They are listed below the attributes and usually include the name, parameters (if any), and return type of the method.

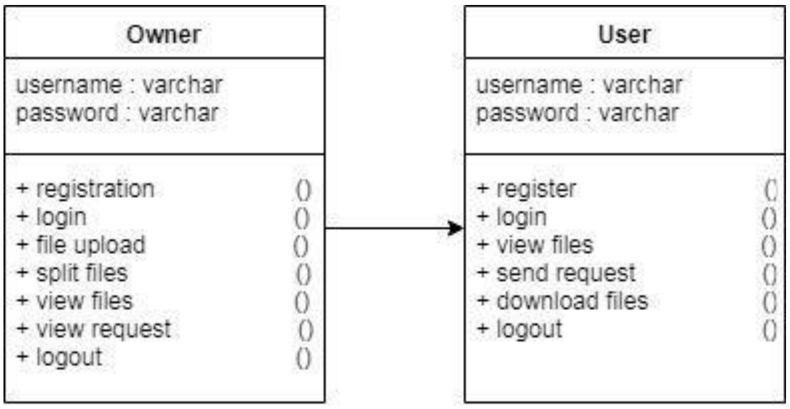
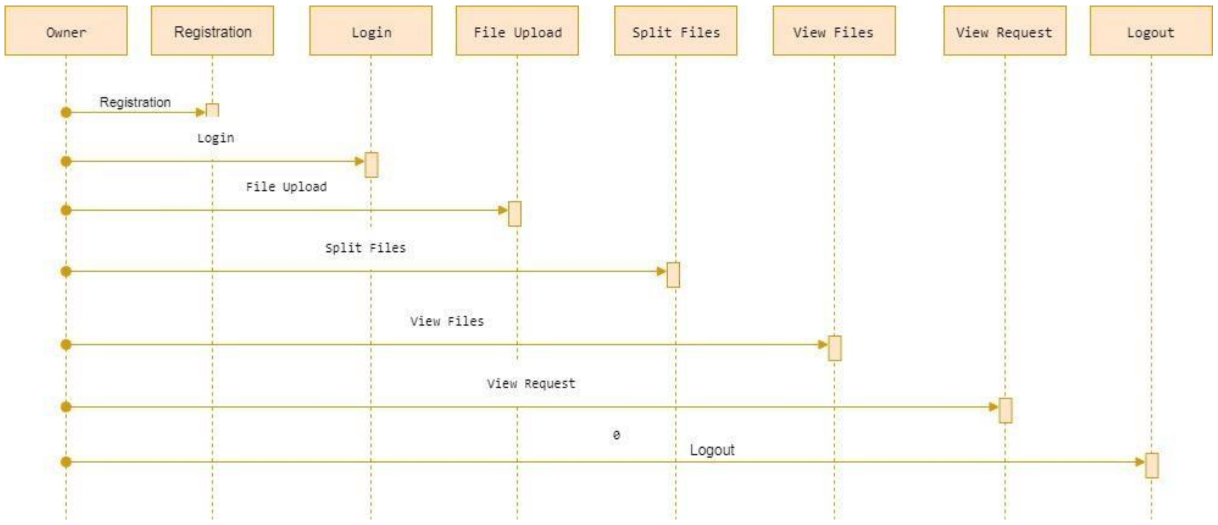


Fig 4: Class Diagram

6.3.3 Sequence Diagram

A sequence diagram is a type of behavioral diagram in UML (Unified Modeling Language) that illustrates the interactions and order of messages exchanged between objects or actors in a system over a specific time period. It shows the flow of communication and the sequence of actions in a scenario, helping to visualize the dynamic behavior of the system. Sequence diagrams are particularly useful for understanding the collaboration and timing of objects in complex scenarios, and they aid in identifying potential issues or improvements in the system's design and functionality.



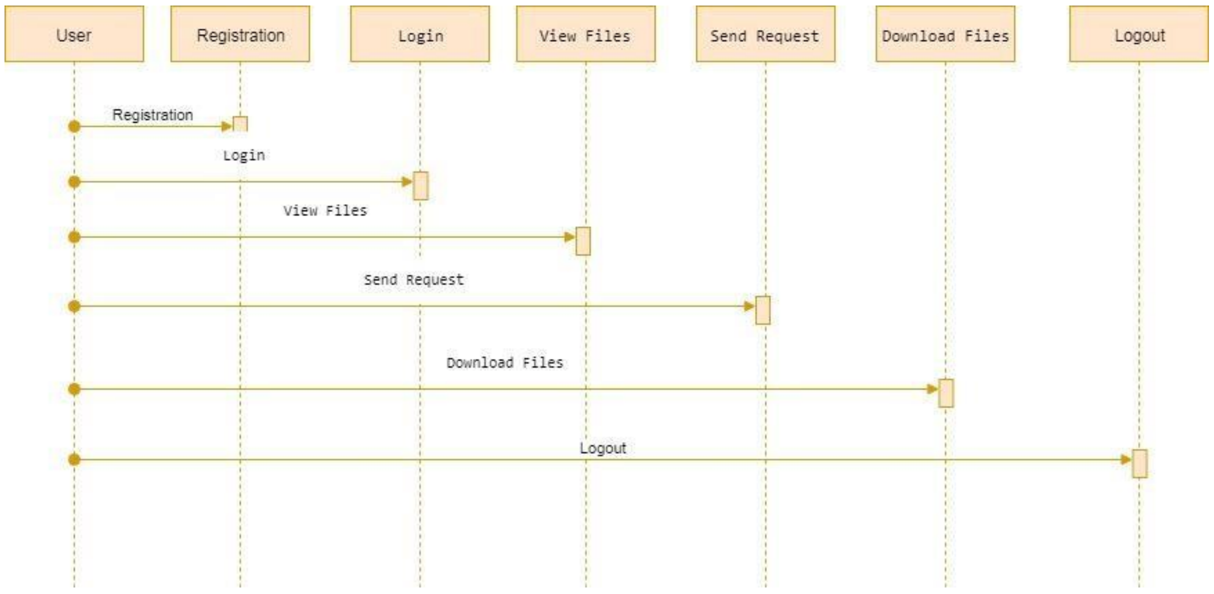


Fig 5: Sequence Diagram

6.3.4 Activity Diagram

An activity diagram is a behavioral diagram in UML (Unified Modeling Language) that depicts the flow of activities or processes within a system. It illustrates the sequential and parallel activities, decision points, and branching paths involved in a specific workflow or use case. Activity diagrams use various shapes, such as nodes for activities, diamonds for decision points, and arrows to represent the flow of control between activities. They provide a visual representation of the overall flow of actions, making it easier to understand complex processes and identify potential bottlenecks or inefficiencies in a system.

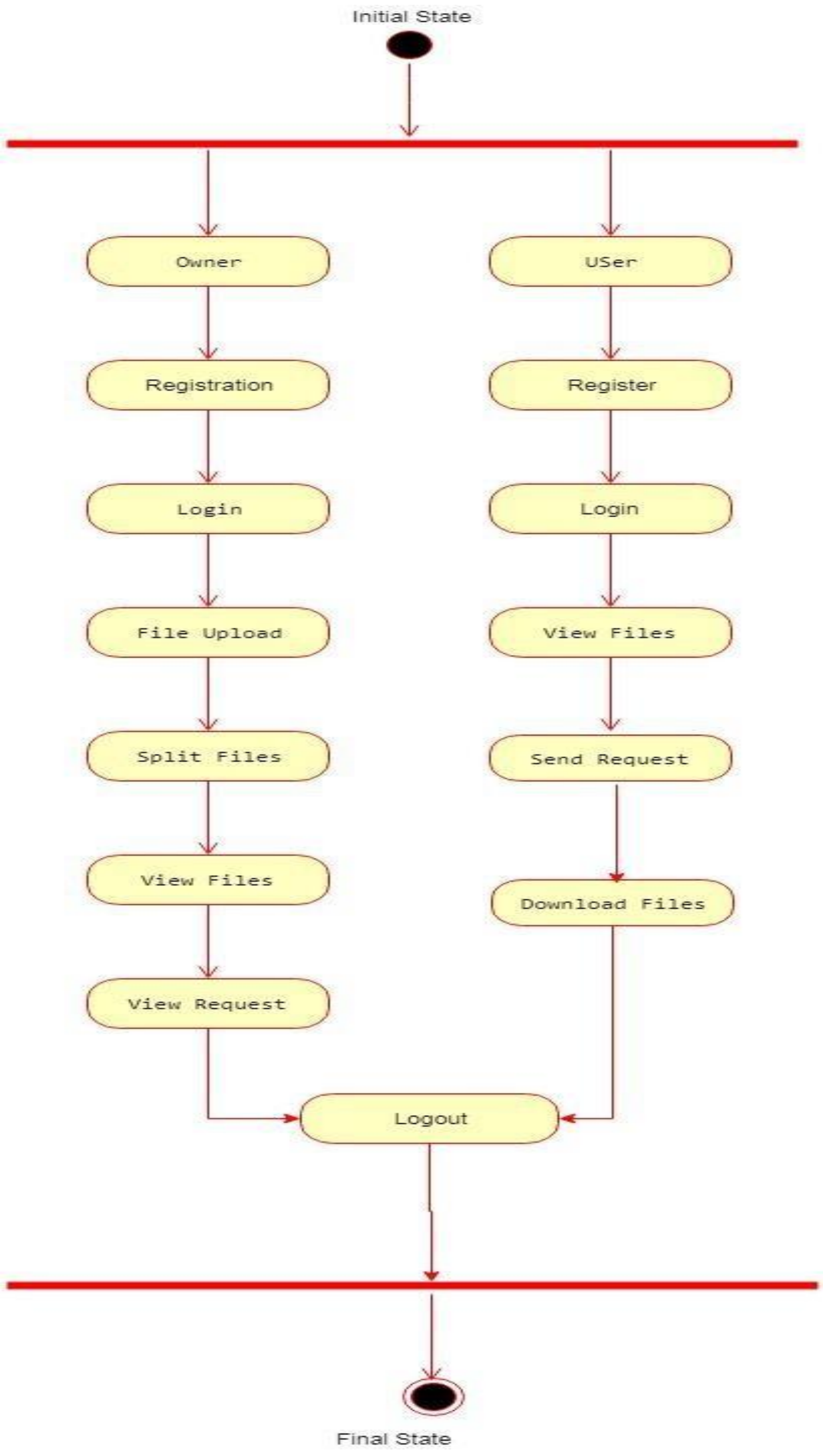


Fig 6: Activity Diagram

7. Multilevel Encryption Analysis

➢ Advance Encryption System

The Advanced Encryption Standard (AES) is a widely used encryption algorithm that is considered highly secure and efficient. It is a symmetric key encryption algorithm, meaning the same key is used for both encryption and decryption. This algorithm was selected by the National Institute of Standards and Technology (NIST) in 2001 as the standard encryption algorithm to replace the older Data Encryption Standard (DES). AES operates on blocks of data and supports key sizes of 128, 192, and 256 bits. It has become the standard encryption algorithm for many applications and industries, including government, finance, and telecommunications. AES provides strong protection for sensitive data and is resistant to various cryptographic attacks, making it a trusted choice for secure communication and data storage.

Here is a simple overview of the AES encryption process:

Key Expansion: The original encryption key is expanded to generate a series of round keys to be used in subsequent encryption rounds.

First round:
A. Round key addition: The input data block (plain text) is XORed with the first round key.

Main round:

a) Subbyte: Each byte of the data block is replaced by the corresponding byte from the AES S-Box, a predefined substitution table.
b) ShiftRows: the bytes of each row of the data block are shifted circularly to the left.
c) MixColumns: Each column of the data block is multiplied by a constant polynomial and a reduced polynomial modulo a predefined irreducible polynomial.
d) AddRoundKey: The result of the previous step is XORed with the corresponding round key.

Final round:
A. sub byte
b. shift line
J. Add round key

The number of main turns depends on the size of the key. 10 rounds for 128- bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

Decryption with AES is basically the reverse process of encryption. The main round is performed in reverse order, and instead of AES S-Box, reverse S-Box is used for the SubBytes stage.

AES is widely accepted due to its security, efficiency and versatility. It is used in various applications such as protecting sensitive data through communication protocols (such as SSL/TLS), file encryption, disk encryption, and other security-related applications.

➢ Data Encryption Standards

The Data Encryption Standard (DES) is a symmetric key encryption algorithm that was widely used in the past for securing sensitive data. Developed in the 1970s, DES operates on 64-bit blocks of data and uses a 56-bit key for encryption and decryption. It employs a substitution-permutation network (SPN) structure to ensure confidentiality and integrity of the data. However, due to advances in technology and computing power, DES is now considered relatively weak and susceptible to brute-force attacks. As a result, it has been largely replaced by more secure encryption algorithms like AES.

Here is an overview of the DES encryption process.

Key generation: A 64-bit encryption key is chosen, but only 56 bits are actually used, the remaining 8 bits are used for parity. The key is transformed and compressed to generate 16 subkeys, each 48 bits long, which are used in the encryption round.
Initial Permutation (IP): 64-bit plaintext is permuted according to fixed table. Feistel structure:
A. Encryption Round: The plaintext is divided into two halves, left (L) and
right (R), each of which is 32 bits long. The encryption process consists of 16 rounds of the same operations performed in L and R.

Extended permutation (E): 32-bit R is extended to 48 bits using constant table. Subkey Mixing: R-enhanced is XORed with a 48-bit subkey derived from the master key.
S-Box permutation: The XOR result is divided into eight 6-bit chunks, each of which is passed through a specific S-Box (permutation box) to produce a 4-bit output.
Sorting: The 32-bit S-Box output is sorted according to a fixed table.

XOR and Swap: The shifted outputs are XORed with the original L and swapped to the new L and R for the next round.
b) Final round: The final round is similar to the encryption round, except that there is no exchange between L and R.
Final Permutation (IP-1): The final result, consisting of the swapped L and R, performs the final permutation to produce the ciphertext.
The decryption process for DES is the same as encryption, but the subkeys are used in reverse order.

➢ RC6 Encryption

RC6 is a symmetric key block cipher encryption algorithm that was developed as an improvement over the widely used Data Encryption Standard (DES). It was designed to provide stronger security and better performance. RC6 operates on blocks of data and supports variable block sizes (typically 32, 64, or 128 bits) and key sizes (up to 2040 bits). It uses a combination of modular arithmetic, bit manipulation, and bitwise XOR operations to perform encryption and decryption. RC6 offers a good balance between security and efficiency, making it suitable for various applications that require secure data transmission and storage. However, it is worth noting that newer encryption algorithms such as AES are generally recommended for modern cryptographic needs.

Key features and characteristics of RC6 encryption include:

Block size: RC6 supports variable block sizes (typically 32, 64, or 128 bits). The block size determines the amount of data processed in each encryption/decryption operation.

Key size: RC6 also supports variable key sizes, typically in the range of 128- 2048 bits. The key size determines the encryption strength and the number of rounds used.

Number of turns: The number of turns in the RC6 depends on the size of the key. The number of rounds used varies, typically 20 rounds for a 128-bit key, and increases linearly with key size.

Key expansion: RC6 uses a key expansion algorithm to derive a set of round keys from the original encryption key. Key expansion processing is done before encryption or decryption.

Encoding/decoding process:
A. Input data is divided into blocks of specified block size.
b) A key expansion algorithm produces a round key.
c) The encryption process involves a series of rounds of manipulation of data blocks and round keys.

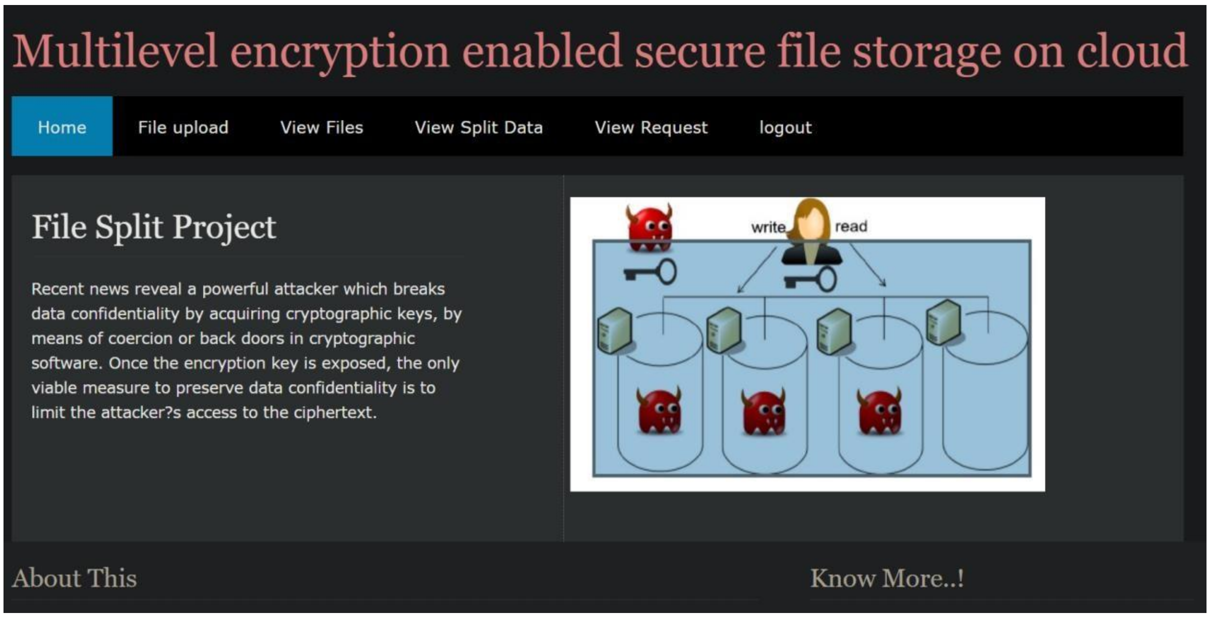
Subkey Mixing: Data blocks are XORed with round keys. Bit shift:
XOR result is subjected to bit rotation operation.
Modular addition: The result of rotation is combined with other round keys using modular addition.
Non-linear blending: The result is subjected to a non-linear blending operation.
d) When all rounds are completed, an encrypted data block is produced as the final output.
e. Decryption follows a similar process but in reverse using the same round key.
RC6 is known for its simplicity and ease of implementation as well as its ability to provide a good level of security. However, it has not reached the same level of adoption as algorithms such as AES due to its relatively slow performance and the availability of more widely standardized algorithms.

- **Project Code :**
  https://github.com/Harshad141/Multilevel-encryption-enabled-secure-file-storage-over-the-cloud

8. Screenshots

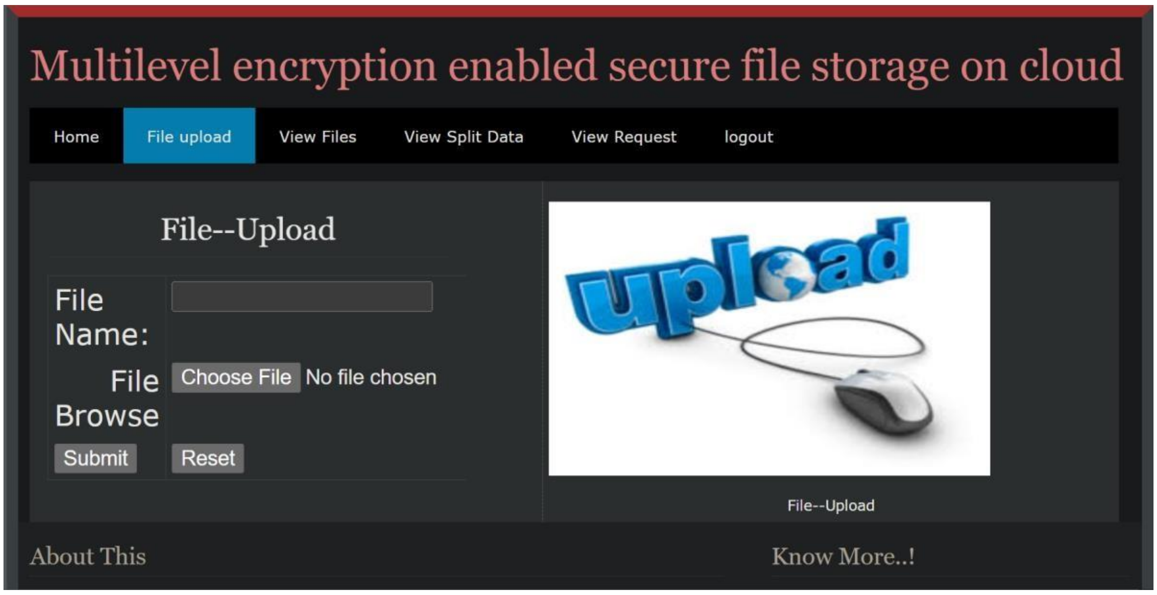
Given below are some screenshots of the project.

(Please note that the below images do not explain the complete working of the project, it's just a demonstration of some parts of the frontend)


Multilevel encryption enabled secure file storage on cloud

File Split Project

Recent news reveal a powerful attacker which breaks data confidentiality by acquiring cryptographic keys, by means of coercion or back doors in cryptographic software. Once the encryption key is exposed, the only viable measure to preserve data confidentiality is to limit the attacker?s access to the ciphertext.



Multilevel encryption enabled secure file storage on cloud

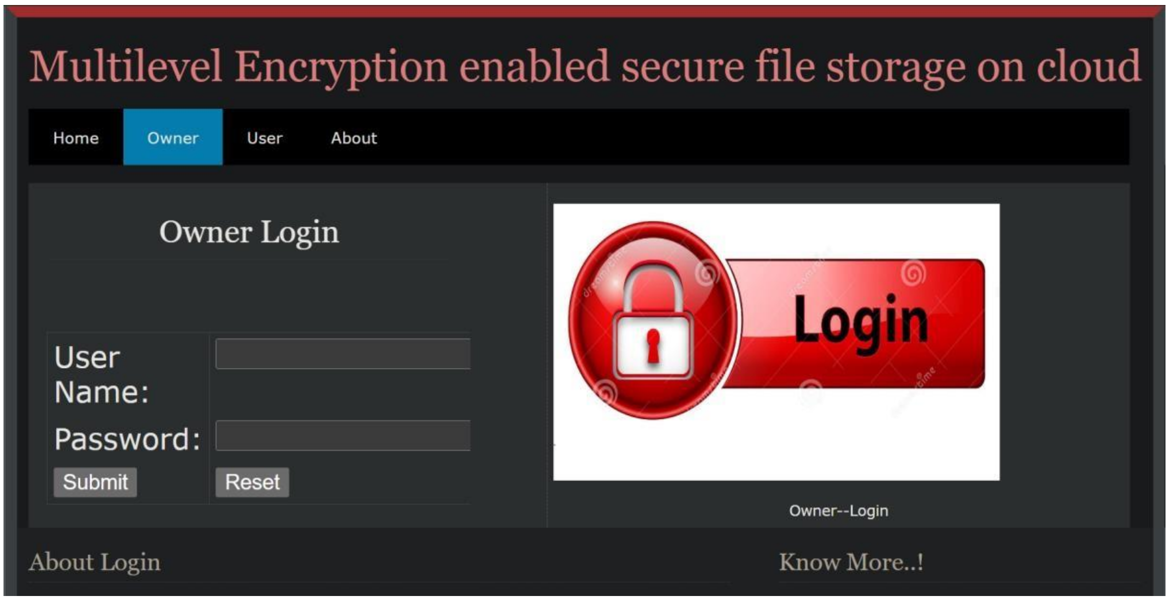
File--Upload

File Name:	
File Browse	Choose File No file chosen
Submit	Reset

File--Upload

About This

Know More..!



Multilevel Encryption enabled secure file storage on cloud

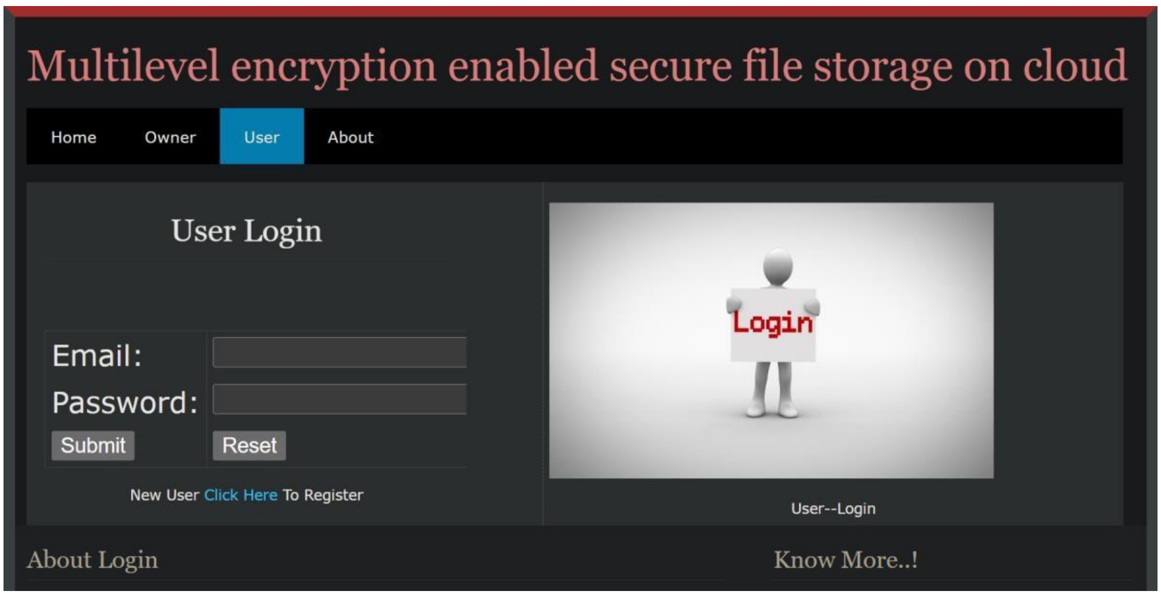
Owner Login

User Name:	
Password:	
Submit	Reset

Owner--Login

About Login

Know More..!



Multilevel encryption enabled secure file storage on cloud

User Login

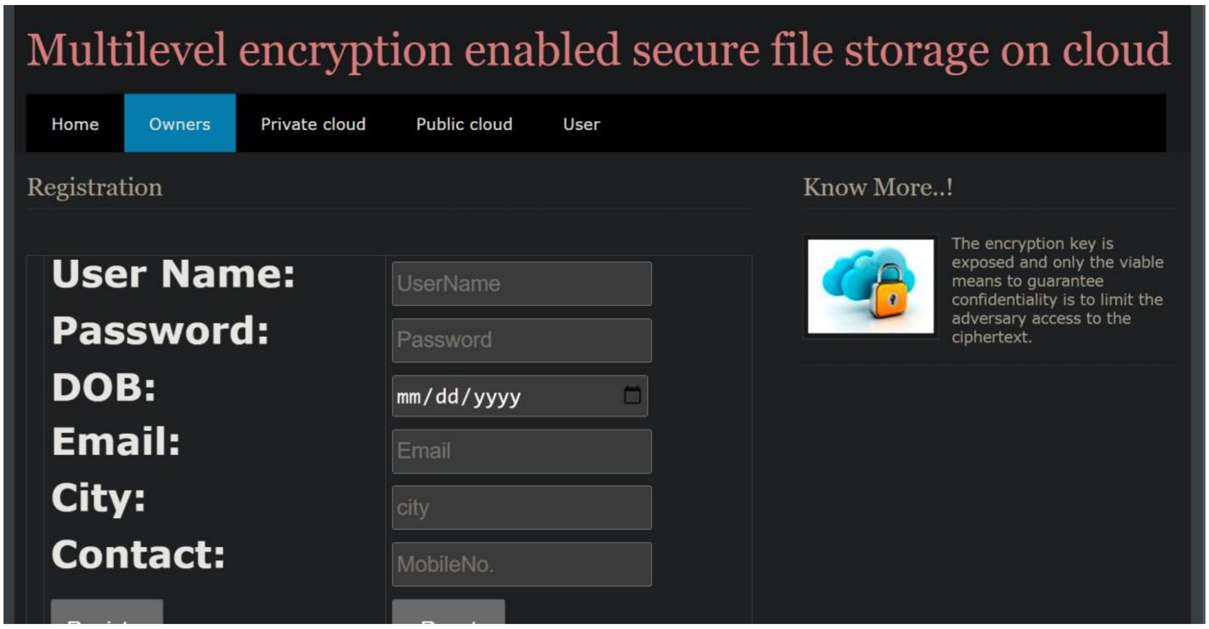
Email:	
Password:	
Submit	Reset

New User Click Here To Register

User--Login

About Login

Know More..!



Multilevel encryption enabled secure file storage on cloud

Home | Owners | Private cloud | Public cloud | User

Registration

User Name:	UserName
Password:	Password
DOB:	mm/dd/yyyy
Email:	Email
City:	city
Contact:	MobileNo.

Know More..!

The encryption key is exposed and only the viable means to guarantee confidentiality is to limit the adversary access to the ciphertext.

9. Future work

Developing this system according to the end-to-end need of the user is not easy. As the system is used, the user's needs continue to change. Potential future improvements of this system include:

• With the advent of technology, the system can be upgraded and adapted to the desired environment.

• Based on future security issues, new technologies such as single sign-on can be used to improve security.

10. Conclusion

The primary objective is to securely store and retrieve data in the cloud, where the data owner lacks direct control. Enhance the protection of your cloud-stored data through the implementation of multilevel encryption techniques. This approach optimizes the storage and retrieval processes within the cloud server. The utilization of specific encryption algorithms further enhances performance during encryption and decryption operations. By adopting this method, data storage and access become more secure and efficient. Our ongoing efforts focus on addressing the challenge of group data sharing within shared data sectors, ensuring that only authorized group members can access the data stored within these sectors.

11. References

[1] Peter Mel and Tim Grace, "The NIST Definition of Cloud Computing", NIST, 2010.

[2] Achill Buhl, "Rising Security Challenges in Cloud Computing", in Proc. of World Congress on Information and correspondence Technologies ,pp. 217-222, Dec. 2011.

[3] Srinivasarao D et al., "Breaking down the Superlative symmetric Cryptosystem Encryption Algorithm", Journal of Global Research in Computer Science, vol. 7, Jul. 2011

[4] Tingyuan Nye and Tang Zhang "An investigation of DES and Blowfish encryption algorithm‟ , in Proc. IEEE Region 10 Conference, pp. 1-4 ,Jan. 2009.

[5] Jitendra Singh Adam et al.," Modified RSA Public Key Cryptosystem Using Short Range Natural Number Algorithm" , International Journal of Advanced Research in Computer Science and Software Engineering ,vol. 2,Aug. 2012.

[6] Manikandan.G et al., "A changed cryptographic plan improving information", Journal of Theoretical and Applied Information Technology, vol. 35, no.2, Jan. 2012.

[7] Niles Maintain and Subhead Bhingarkar, " The examination and Judgment of Nimbus, Open Nebula and Eucalyptus", International Journal of Computational Biology , vol. 3, issue 1, pp 44-47, 2012.