

A

Project Report on

**Web based application for secure file storage on local host using
hybrid cryptography**

Submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

**Juhi Mohta
20EG105221**

**Pagidoju Shivani
20EG105237**

**Koganti Hemasri
20EG105240**

Under the guidance of

**Dr. V. Subrahmanyam
Associate Professor**



Department of Computer Science and Engineering

Venkatapur(V), Ghatkesar(M), Medchal(D)., T.S- 500088

2023-2024

CERTIFICATE

This is to certify that the project report entitled “**Web based application for secure file storage on local host using hybrid cryptography**” that is being submitted by **Juhi Mohta** bearing hall ticket number **20EG105221**, **P. Shivani** bearing hall ticket number **20EG105237** and **K. Hemasri** bearing hall ticket number **20EG105240**, in partial fulfilment for the award of **Bachelor of Technology in Computer Science and Engineering** to the **Anurag University** is a record of bonafide work carried out by them under my guidance and supervision from academic year 2023 to 2024.

The result presented in this project has been verified and found to be satisfactory. The results embodied in this Report have not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Guide

Dr. V. Subrahmanyam
Associate Professor, CSE

Dean, CSE

Dr. G. Vishnu Murthy

External Examiner

DECLARATION

We hereby declare that the report entitled **Web based application for secure file storage on local host using hybrid cryptography** submitted in partial fulfilment of the requirements for the award of **Bachelor of Technology in Computer Science and Engineering** from Anurag University is a record of an original work done by us under the guidance of **Dr. V. Subrahmanyam, Associate Professor, Department of CSE** and this report has not been submitted to any other University or institution for the award of any degree or diploma.

Juhi Mohta
20EG105221

Pagidoju Shivani
20EG105237

Koganti Hemasri
20EG105240

Place: Anurag University, Hyderabad

Date:

ACKNOWLEDGMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr. V. Subrahmanyam**, Associate Professor, Department of Computer Science and Engineering for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B.Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Dept. of CSE, Anurag University. We also express our deep sense of gratitude to **Dr. V V S S Balaram**, Academic Coordinator, and **Dr. Pallam Ravi**, Project in-charge, **Dr. V. Rama Krishna**, Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

Juhi Mohta
20EG105221

Pagidoju Shivani
20EG105237

Koganti Hemasri
20EG105240

ABSTRACT

In today's digital age, the security of sensitive information and data is of paramount importance. As data breaches and cyberattacks continue to escalate, ensuring secure file storage on a local host is a critical concern. This project aims to address this issue through the implementation of hybrid cryptography techniques to protect files stored on a local host. Hybrid cryptography combines the strengths of both symmetric and asymmetric encryption to provide a robust security solution. Symmetric encryption is efficient and fast, while asymmetric encryption offers strong key management and distribution. By combining these two methods, this project provides a secure and efficient approach to safeguarding local file storage. But sometimes a single technique or algorithm alone cannot provide high-level security. So, we have introduced a new security mechanism that uses a combination of multiple cryptographic algorithms of symmetric keys. All the algorithms use 128-bit keys

Key information will contain the information regarding the encrypted part of the file, the algorithm, and the key for the algorithm. File during encryption is split into three parts. These individual parts of the file will be encrypted using different encryption algorithms simultaneously with the help of a multithreading technique.

CONTENTS

| S. No. | PAGE NO |
|---------------------------|----------------|
| 1. Introduction | 1 |
| 1.1. Overview | 2 |
| 1.2. Problem Statement | 2 |
| 1.3. Objective | 3 |
| 1.4 Project Scope | 4 |
| 2. Literature Survey | 6 |
| 3. Analysis | 8 |
| 3.1 Existing System | 8 |
| 3.2 Proposed System | 9 |
| 3.3 Feasibility Study | 10 |
| 3.4 Software Requirements | 11 |
| 3.5 Module Description | 12 |
| 3.6 Details | 15 |
| 4. System Design | 19 |
| 4.1. System Analysis | 19 |
| 4.1.1 System Architecture | 22 |
| 4.2. UML Diagram | 24 |
| 4.2.1 Sequential Diagram | 25 |
| 4.2.2 Activity Diagram | 26 |
| 4.2.3 Use case Diagram | 26 |
| 4.2.4 Class Diagram | 27 |
| 4.3 System Design | 28 |

| | |
|-------------------------------|----|
| 4.3.1 Modular Design | 28 |
| 4.3.2 Database Design | 30 |
| 5. Implementation | 32 |
| 5.1 Working of AES | 34 |
| 5.2 Working of RSA | 35 |
| 6. Testing | 43 |
| 6.1 Software Testing | 43 |
| 6.1.1 Unit Testing | 43 |
| 6.1.2 Integration Testing | 44 |
| 6.1.3 User Acceptance Testing | 44 |
| 6.1.4 Validation Checking | 45 |
| 6.1.5 Output Testing | 46 |
| 6.2 Plagiarism Report | 47 |
| 6.3 Test Cases | 48 |
| 7. Results | 49 |
| 8. Future Scope | 54 |
| 9. References | 56 |
| 10. Appendices | 58 |

LIST OF FIGURES

| FIGURE NO | TITLE | PAGE NUMBER |
|-----------|----------------------------|-------------|
| 4.1.1 | Block Diagram | 24 |
| 4.2.1 | Sequential Diagram | 25 |
| 4.2.2 | Activity Diagram | 26 |
| 4.2.4 | Class Diagram | 28 |
| 4.3.1 | Dataflow diagram | 30 |
| 5.1 | Working with AES Algorithm | 35 |
| 5.1.1 | Working with RSA Algorithm | 36 |
| 6.1.5 | Types of Software Testing | 47 |
| 6.2 | Plagiarism Report | 47 |
| 7.1 | Running Sever | 49 |
| 7.2 | Signup Page | 49 |
| 7.3 | Login Page | 50 |
| 7.4 | Details of Registered User | 50 |
| 7.5 | File Upload page | 51 |
| 7.6 | View Uploaded files | 51 |
| 7.7 | Authentication page | 52 |
| 7.8 | OTP in mail | 52 |
| 7.9 | Authentication page | 52 |

LIST OF TABLES

| TABLE NO | TITLE | PAGE NUMBER |
|----------|------------|-------------|
| 6.3 | Test cases | 48 |

In today's hyper-connected digital landscape, safeguarding sensitive data and information stands as an imperative mission. With the proliferation of cyberattacks and data breaches, the need for fortified defences to ensure secure file storage on local hosts has never been more pressing. This project emerges as a beacon of resilience in the face of such challenges, harnessing the formidable power of hybrid cryptography techniques to fortify the digital bastions against malicious incursions. In the crucible of security innovation, hybrid cryptography emerges as a formidable ally, seamlessly blending the unparalleled strengths of symmetric and asymmetric encryption.

Asymmetric encryption, with its robust key management and distribution capabilities, lays the foundation for a secure infrastructure, while symmetric encryption, renowned for its speed and efficacy, stands poised to safeguard data with unwavering precision. By harmonizing these two cryptographic paradigms, this project not only pioneers a path towards impregnable data fortification but also heralds a new era of confidence and trust in local file storage security. At its core, this endeavour represents more than just a technical solution; it embodies a philosophy of resilience and adaptability, poised to withstand the relentless onslaught of digital threats. In the ever-evolving landscape of cybersecurity, where threats morph and evolve with alarming agility, the need for innovative solutions becomes paramount. This project, a testament to ingenuity and foresight, stands as a bastion against the tide of digital malevolence. With each line of code meticulously crafted and every encryption layer fortified, it embodies a relentless pursuit of excellence in safeguarding the digital realm. Furthermore, by incorporating cutting-edge technologies such as machine learning algorithms for anomaly detection and behaviour analysis, this project elevates the standards of protection, providing a proactive defence mechanism against emerging threats. By offering a reliable and effective solution for securing local file storage, this project transcends the boundaries of mere encryption protocols, ushering in a new era of digital sovereignty and empowerment. As organizations and individuals alike navigate the treacherous waters of cyberspace, this project stands as a stalwart guardian, steadfast in its commitment to preserving the sanctity of sensitive information in the modern digital age.

1.1 OVERVIEW

In the realm of cryptography, AES (Advanced Encryption Standard) stands tall as a cornerstone of modern encryption protocols. Renowned for its efficiency, AES employs symmetric encryption techniques, allowing for rapid data encryption and decryption while minimizing resource usage. Its widespread adoption in various applications underscores its reliability and effectiveness in safeguarding sensitive information.

On the other hand, RSA (Rivest-Shamir-Adleman) encryption represents a stalwart in the domain of asymmetric cryptography. Leveraging the mathematical intricacies of prime factorization, RSA offers a robust solution for key management, enabling secure key distribution and authentication in communication channels. Despite its computational overhead, RSA remains indispensable in scenarios where stringent security measures are paramount.

By combining the virtues of AES's efficiency and RSA's robust key management capabilities, hybrid cryptography emerges as a formidable ally in the quest for secure local file storage. The marriage of symmetric and asymmetric encryption techniques not only addresses the shortcomings of each approach but also synergizes their strengths to offer a comprehensive solution that prioritizes both security and efficiency. In addition to AES and RSA, elliptic curve cryptography (ECC) contributes to hybrid cryptography, enhancing security with smaller key sizes, ideal for resource-constrained environments, further fortifying local file storage against cyber threats.

1.2 PROBLEM STATEMENT

In an era marked by the relentless march of digitalization, the critical need for secure file storage solutions becomes increasingly apparent, especially concerning sensitive or confidential data housed on local machines. Conventional encryption methods, while serving as initial bulwarks, often reveal their limitations, presenting challenges in terms of both security and user-friendliness.

This project ambitiously aims to confront and surmount these obstacles by pioneering a system for secure file storage on a local host, ingeniously leveraging the power of

hybrid cryptography. In the ever-evolving landscape of data security, traditional file storage systems remain susceptible to an array of threats, ranging from the spectre of unauthorized access to the ominous spectre of data breaches and malicious intrusions. To effectively fortify the confidentiality and integrity of stored files against such adversities, the imperative of implementing an advanced security mechanism becomes unequivocal.

However, in the pursuit of such robust security measures, it is paramount to maintain a delicate equilibrium between fortitude and usability. Overly stringent encryption protocols, while ostensibly enhancing security, can inadvertently engender complications for users, particularly in the realm of managing encryption keys.

The spectre of potential data loss looms ominously should these keys be forgotten or misplaced, underscoring the necessity of integrating user-friendly mechanisms within the security framework. Through the meticulous implementation of this cutting-edge secure file storage system, the project not only endeavours to bolster data security practices at the local level but also strives to engender a paradigm shift towards a more secure and user-centric digital ecosystem.

1.3 OBJECTIVE

In an era, fraught with escalating cybersecurity threats and rampant data breaches, the imperative for safeguarding sensitive and confidential information stored on personal or organizational local hosts has never been more pronounced. Recognizing this urgent need, the project endeavours to develop a robust solution grounded in hybrid cryptography techniques to address the critical requirement for secure file storage on local hosts.

At its core, this initiative seeks to amalgamate the formidable strengths of both symmetric and asymmetric cryptography, harnessing their combined prowess to fortify data confidentiality and integrity. By leveraging a hybrid approach to encryption, the system aims to erect an impregnable defence against unauthorized access and data theft, assuring users that their locally stored files remain cocooned from potential threats.

The project is poised to confront the multifaceted challenges and intricacies inherent in implementing hybrid cryptography for secure file storage on local hosts. Foremost among these challenges is the imperative to uphold system performance integrity throughout the encryption and decryption processes, ensuring that users can access their files expeditiously and without undue delay.

Moreover, the establishment of a robust and user-friendly key management system emerges as a pivotal consideration. This system must enable users to securely store and retrieve encryption keys, thus bolstering the overall security architecture while preserving ease of access and usability. Scalability represents another paramount concern, necessitating a solution that can seamlessly adapt to accommodate an array of file types and sizes without compromising on performance or security standards.

The project thus endeavours to strike a delicate balance between versatility and efficacy, ensuring that users can entrust their data to the system with unwavering confidence. Furthermore, the implementation of comprehensive error handling and data recovery mechanisms assumes heightened significance. These mechanisms are indispensable in averting potential data loss or corruption, thereby cementing the system's reputation as a reliable guardian of digital assets. In sum, the project represents a concerted effort to surmount the formidable challenges posed by contemporary cybersecurity landscapes.

Through the judicious fusion of cryptographic techniques, meticulous attention to performance optimization, and a steadfast commitment to user-centric design principles, it aspires to carve a niche as a preeminent safeguard for secure file storage on local hosts.

1.4 PROJECT SCOPE

The project "Secure File Storage using Hybrid Cryptography" is a cybersecurity initiative designed to ensure the confidentiality and integrity of digital files. It combines two cryptographic techniques, symmetric and asymmetric cryptography, to safeguard data. This type of project falls under the category of Information Security and Encryption.

It aims to provide a robust and versatile solution for storing sensitive information securely, making it suitable for applications in various domains, including data privacy, cloud storage, and secure communication. The project involves encryption and decryption processes, key management, and authentication mechanisms to protect files from unauthorized access and tampering.

By melding the strengths of symmetric and asymmetric cryptography, the project engenders a holistic defence mechanism that transcends the limitations of individual cryptographic methods.

This synergistic approach not only enhances the robustness of data protection but also fosters a versatile ecosystem capable of accommodating the evolving needs of modern cybersecurity landscapes. As a testament to its efficacy, the Secure File Storage project stands as a beacon of innovation, offering a pragmatic solution to the perennial challenge of safeguarding digital assets.

Its implications extend far beyond mere encryption protocols, embodying a paradigm shift in how organizations perceive and implement data security strategies.

With its comprehensive suite of protective measures, this initiative heralds a new era of confidence and trust in the digital realm, empowering stakeholders to navigate the complex terrain of information security with poise and assurance.

Moreover, through continuous refinement and adaptation to emerging threats, the project serves as a dynamic force in the ongoing battle against cyber vulnerabilities, ensuring that data remains shielded from adversaries and instilling a sense of resilience within the cybersecurity ecosystem.

A literature review is nothing but an objective, aim, or summary of whatever research has been done relevant to a certain topic. The following published articles have been referred to create a base for my project. The literature review serves as a comprehensive examination of existing research pertinent to the project's focus. The referenced articles provide a foundation for understanding the landscape surrounding the chosen topic, offering insights, methodologies, and findings that contribute to shaping the project's objectives and methodologies. Following are some papers been referred to -

1. Author – M. Malarvizhi, J. Angela JennifaSujana, T. Revathi, Year – 2014
Description - The main focus of the paper is on the integrity of files and restoring the files if integrity is violated. The proposed system uses a pattern of each protected file to determine its modification. The method used for pattern generation is cryptographic hash functions. The system also uses a database that stores the files that need to be protected and their hash codes. To check the integrity of the file the hash code of the file is produced and checked with one in the database. If the file is successfully tested positively then access is granted otherwise the administrator gets alerted and if is saved copy is available of the same file then the file is restored. A new approach to user authentication using a digital signature.
2. Author - Jerzy Kaczmarek, MichałWróbel, Year 2008 Description – This paper describes an approach to the integrity of files and restoring the files if any problem arises in the future. This proposed course uses a pattern of each protected file to determine its modification. Methods used for pattern generation are cryptographic hash functions. This system uses a database that stores the names of all files that are to be protected and their hash codes. To check the integrity of the file the hash code of the file is produced and checked with one in the database. After the file is verified only access is granted the administrator is been alerted about the problems and a saved copy of the same file is restored safely. Secure file sharing using cryptographic techniques.

3. Author - Tulip Dutta, Amarjyoti Pathak, Year – 2016 Description – This paper discusses how a secret key can be shared with other users to whom access needs to be given. It discusses the problem of using a single key to encrypt all data and using different keys for different files. The solution described in the paper tries to address both the problem using key aggregation. In key aggregation, different data files are encrypted with different keys, and then for decryption, a single aggregated key is used. The encryption algorithm used is AES and the system is being implemented in Java using the key store data structure. Achieving cloud security using third-party auditor, MD5, and identity-based encryption.
4. Author - Bilal Habib, Bertrand Cambou, Duane Booher, Christopher Philabaum, Year – 2017 Description – This paper provides a new method to implement the public key infrastructure. The PKI has the disadvantage that the mathematical relation between public and private between the public and the private key is maintained. The paper proposes a new PKI scheme with addressable elements (PKA). The approach proposed removes the mathematical relation between public and private keys using addressable cryptographic tables. Secure data sharing in cloud storage using key aggregation cryptography.
5. Author -Mr. Rohit Barvekar, Mr. ShrajalBehere, Mr. Yash Pounikar, Ms. Anushka Gulhane, Year 2018 Description - The proposed security mechanisms will prevent confidential data from being misused making the system more reliable. High speed: The proposed method will make encryption and decryption with proper keys much faster than usual. Security in Cloud Computing using Cryptographic Algorithms.

3.1 EXISTING SYSTEM

The existing system for secure file storage on a local host likely relies on conventional encryption methods or simple file access controls. It may involve basic password protection, file permissions, or software that provides basic encryption. The existing system might use symmetric encryption methods, where a single encryption key is used for both encryption and decryption.

This key may be stored locally on the host, posing a risk if the host is compromised. Since only a single cryptography algorithm was used, it did not provide enough security. For example, if only AES algorithm was used, there was high speed but no security, if only RSA was used, there was high security but less speed. User access is usually controlled by basic username and password authentication, which can be vulnerable to password-guessing attacks or breaches.

The prevailing system for secure file storage on a local host typically relies on conventional encryption methods or rudimentary file access controls, often entailing basic password protection or simplistic encryption software. Such systems may lean heavily on symmetric encryption methods, where a single encryption key handles both encryption and decryption tasks, thus introducing vulnerabilities if the key is compromised. This singular reliance on a single cryptographic algorithm, be it AES for speed or RSA for security, renders the system susceptible to sophisticated attacks.

User access, typically governed by rudimentary username and password authentication, remains vulnerable to password-guessing exploits or breaches, underscoring the urgent need for more robust security measures. Additionally, the system's resistance to advanced attacks is questionable, as the theft of encryption keys or access credentials could jeopardize the security of stored files.

The limitations of the existing system highlight the necessity for a more sophisticated approach to secure file storage on local hosts. By integrating hybrid cryptography techniques, leveraging the strengths of both symmetric and asymmetric encryption, the proposed solution aims to address these vulnerabilities effectively. Through enhanced encryption protocols, rigorous key management, and advanced authentication mechanisms, the project endeavors to fortify the security posture of

local file storage systems, offering comprehensive protection against a myriad of cyber threats. Such an evolution in security measures promises not only enhanced data protection but also instills greater confidence in the integrity of stored information, safeguarding against the ever-evolving landscape of cyber threats.

3.2 PROPOSED SYSTEM

The proposed system for secure file storage on a local host represents a paradigm shift, harnessing the potent synergy of hybrid cryptography to fortify data protection. By melding the swiftness of AES with the robust security of RSA, it promises a formidable defence against cyber threats. Enhanced user access control mechanisms, including robust multi-factor authentication like biometric verification, further elevate the system's resilience against unauthorized access attempts. This comprehensive fusion of cryptographic techniques not only ensures the confidentiality and integrity of stored files but also instils a newfound sense of confidence in the security posture of local file storage environments.

Moreover, the system will implement dynamic key management strategies to mitigate the risk of key compromise or theft, ensuring the ongoing integrity of encryption processes. Through continuous monitoring and updates, it pledges unwavering vigilance against emerging threats.

- **Symmetric Encryption:** Employing advanced encryption techniques such as AES, every file will be securely encrypted with a unique symmetric key, ensuring robust protection against unauthorized access and data breaches.
- **Asymmetric Encryption:** Asymmetric encryption will further protect the symmetric keys that are used for file encryption. The public keys will be applied to cipher the symmetric keys, and each user will receive their own set of private and public keys. By implementing these features and leveraging hybrid cryptography, the proposed system will provide a much higher level of security for file storage on a local host, making it significantly more resistant to various forms of attacks and data breaches. A secure backup and recovery mechanism will be established to ensure that data can be restored in the event of hardware failures or data corruption.

By integrating these features and harnessing the power of hybrid cryptography, the proposed system will not only offer a significantly higher level of security for file storage on a local host but also bolster its defences against an array of cyber threats and potential data breaches. Additionally, a comprehensive secure backup and recovery mechanism will be instituted, guaranteeing data integrity and resilience in the face of hardware failures or data corruption. Additionally, a comprehensive secure backup and recovery mechanism will be instituted, guaranteeing data integrity and resilience in the face of hardware failures or data corruption. This holistic approach to data protection ensures that organizations and individuals can trust in the reliability and robustness of their local file storage solutions, fostering a safer digital environment for sensitive information.

3.3 FEASIBILITY STUDY

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resources and time. The different feasibilities that have to be analysed are

- **Operational Feasibility:** Operational Feasibility deals with the study of the prospects of the system to be developed. This system operationally eliminates all the tensions of the admin and helps him in effectively tracking the project's progress. This assessment considers factors such as the system's compatibility with current workflows, the availability of necessary resources, and the ease of adoption by end-users. In the context of the proposed system, operational feasibility would examine its ability to streamline administrative tasks and facilitate efficient project tracking. It would assess whether the system addresses existing pain points for administrators, such as cumbersome manual processes or inadequate monitoring capabilities. By operationalizing tasks and providing comprehensive tracking functionalities, the system aims to alleviate administrative burdens and enhance productivity. Ultimately, operational feasibility seeks to ensure that the proposed system can be seamlessly integrated into the organization's operations, maximizing

its potential benefits while minimizing disruptions and challenges for stakeholders.

- **Economic Feasibility:** It is an assessment of the economic justification for a computer-based project. As hardware was installed from the beginning; for lots of purposes the cost of the project of hardware is low. Since the system is network-based, any number of employees connected to the LAN within that organization can use this tool at any time. Furthermore, the network-based nature of the system enhances its accessibility and scalability, allowing any number of employees connected to the LAN to utilize the tool concurrently. This not only maximizes resource utilization but also minimizes additional expenses associated with hardware procurement or software licensing for individual users. By leveraging existing infrastructure and facilitating widespread user access, the proposed system demonstrates strong operational feasibility, promising efficient resource utilization and economic benefits for the organization.
- **Technical Feasibility:** It is the assessment of the technical resources of the organization. The system is technically feasible for development and can be developed with the existing facility. It assesses the project's capability to be developed using existing technology and infrastructure. It examines whether the proposed solution can be implemented within technological constraints, considering factors such as compatibility, scalability, and resource availability. Technical feasibility also evaluates the project team's expertise and the availability of necessary technical resources. It ensures that the proposed solution aligns with industry standards and can be effectively integrated with existing systems. This assessment informs stakeholders about the project's technical viability and potential challenges that may arise during implementation.

3.4 SYSTEM REQUIREMENTS

Software Requirements

- Operating system: Windows Family.
- Coding Language: J2EE (JSP,Servlet,Java Bean)
- Data Base: mySQL.

- Web Server: Wamp server
- Framework: Django and python

Hardware Requirements

- System: Pentium IV 3.4 GHz.
- Hard Disk: 40 GB.
- Floppy Drive: 1.44 Mb.
- Monitor: 14' Colour Monitor.
- Ram: 4 GB.

By meticulously adhering to these system requirements (software requirements and hardware requirements), the web-based application built with Django ensures a resilient, adaptable, and safeguarded environment, ideal for reliable secure file storage and seamless user experience.

3.5 MODULE DESCRIPTION

In programming, modules refer to self-contained units of code that encapsulate specific functionalities. They serve to organize code into manageable and reusable components, promoting modularity, maintainability, and code reusability. Modules can contain variables, functions, classes, or other programming constructs, and they facilitate code organization, collaboration among developers, and the building of larger, more complex software systems.

It requires different modules or components to work together to create a secure file system for storage on a local host that uses hybrid cryptography to maintain the privacy and authenticity of stored files. This entails integrating various modules or components harmoniously to establish a secure file system for local host storage. Hybrid cryptography ensures the privacy and authenticity of stored files through collaborative encryption techniques. Below are a few models for this project.

1. User Authentication Module

This module is Responsible for authenticating users before granting access to the system. May use techniques such as password-based authentication or multi-factor

authentication. Ensures only authorized users can interact with the file storage system. The Authentication module is fundamental for verifying user identities prior to granting access to the system. It employs various techniques like password-based authentication or multi-factor authentication to validate user credentials. By implementing robust authentication protocols, it safeguards against unauthorized access, ensuring that only authenticated and authorized users can interact with the file storage system. This module plays a pivotal role in bolstering the security of the system, establishing trust in user interactions, and fortifying the overall integrity of the stored data.

2. File Upload and Download Module

Allows users to upload files securely to the system. Provides options for downloading and accessing stored files. May include features like file versioning and metadata storage. The File Management module facilitates secure file upload, ensuring that users can transfer data to the system without compromising confidentiality or integrity. It offers seamless options for downloading and accessing stored files, providing a user-friendly interface for efficient retrieval. Additionally, the module may incorporate advanced features such as file versioning, enabling users to track and manage multiple iterations of documents effectively. Moreover, metadata storage capabilities enhance file organization and searchability, empowering users to categorize and retrieve files based on relevant attributes. Together, these functionalities optimize the user experience, streamline file management processes, and reinforce the system's resilience against potential security threats.

3. Hybrid Cryptography Module

Combines symmetric and asymmetric encryption for secure file storage. Generates and manages encryption keys. Encrypts files using symmetric keys, which are in turn encrypted with the recipient's public key. Decrypts files using the recipient's private key and the stored symmetric key. The Encryption module forms the backbone of secure file storage by integrating symmetric and asymmetric encryption techniques. It operates by generating and managing encryption keys, a pivotal aspect of data

security. Utilizing symmetric encryption, files are encrypted with a single, shared key, known as the Advanced Encryption Standard (AES), renowned for its efficiency and speed in cryptographic operations. Subsequently, the module enhances security by encrypting the symmetric key itself using the recipient's public key, a process facilitated by asymmetric encryption, such as the Rivest-Shamir-Adleman (RSA) algorithm. This double-layered encryption approach ensures that even if the encrypted file is intercepted, the content remains protected by the recipient's private key, which is required for decryption. Upon retrieval, the module decrypts the symmetric key using the recipient's private key and then decrypts the file content using the decrypted symmetric key. By seamlessly blending AES and RSA encryption mechanisms, the module delivers robust protection against unauthorized access and data breaches, ensuring the confidentiality and integrity of stored files in the local host storage system.

4. Key Management Module

Manages the creation, distribution, and storage of encryption keys. Ensures keys are securely stored and accessible only to authorized users.

Provides key rotation and revocation mechanisms for enhanced security. The Key Management module plays a critical role in the secure file storage system by overseeing the entire lifecycle of encryption keys. It begins with the creation of encryption keys, a process meticulously managed to generate strong and unique keys for each file encryption session. These keys are then distributed to authorized users or entities, ensuring that only those with the requisite permissions can access encrypted files. A key aspect of the module is the secure storage of encryption keys, safeguarding them against unauthorized access or compromise. Access controls and encryption mechanisms are implemented to ensure that keys are accessible only to authorized users and are protected from external threats. Moreover, the module incorporates key rotation and revocation mechanisms to bolster security further. Key rotation involves periodically changing encryption keys to mitigate the risk associated with long-term key exposure. Meanwhile, key revocation enables the invalidation of compromised or no longer authorized keys, preventing unauthorized access to encrypted data. By meticulously managing encryption keys throughout their lifecycle, the Key Management module ensures the integrity, confidentiality, and availability of data within the secure file storage system, reinforcing its resilience against potential security threats.

5. Access Control Module

The Access Control module is essential for maintaining the integrity and confidentiality of files within the secure storage system. It defines and enforces access control policies that dictate who can access, modify, or delete files, ensuring that sensitive data remains protected from unauthorized manipulation or disclosure. This module empowers users to specify access permissions for their files, granting granular control over file operations based on user identity or predefined criteria. Users can designate individuals or groups with specific access privileges, such as read, write, or delete permissions, tailoring access levels to meet their unique security requirements. Defines and enforces access control policies for files. Allows users to specify who can access, modify, or delete their files. May integrate role-based access control (RBAC) for more complex permission structures. In more complex scenarios, the Access Control module may integrate role-based access control (RBAC) mechanisms. RBAC assigns permissions based on predefined roles, streamlining access management by grouping users with similar responsibilities or job functions. This hierarchical approach simplifies administration and enhances security by reducing the potential for access errors or unauthorized data exposure. By enforcing robust access control policies and integrating RBAC where necessary, the module ensures that only authorized users can interact with files according to predefined permissions, bolstering the overall security posture of the file storage system.

3.6 DETAILS

The project aims to develop a secure file storage system on a local host using hybrid cryptography. This system will ensure the confidentiality and integrity of stored files.

Impact on Environment

The environmental impact of a secure file storage project using hybrid cryptography may be relatively low compared to many other types of projects. However, it's essential to consider the following:

- **Energy Consumption:** Increased energy consumption due to server operation, especially if a dedicated server is used. It is a

critical consideration for the secure file storage system, particularly regarding server operation. Utilizing a dedicated server may lead to increased energy consumption, as the server needs to remain operational to facilitate file storage and access. This heightened energy usage not only impacts operational costs but also contributes to environmental concerns, particularly in data centre environments. Therefore, the project will explore strategies to optimize energy efficiency, such as server consolidation, virtualization, and implementing energy-saving technologies to mitigate environmental impact and operational expenses.

- **Carbon Footprint:** Estimate the carbon emissions from energy usage and server operations. Assessing the project's carbon footprint involves estimating the carbon emissions associated with energy consumption and server operations. This entails calculating the amount of greenhouse gases emitted during the production and utilization of energy for running servers. By quantifying carbon emissions, the project can evaluate its environmental impact and explore mitigation strategies to minimize carbon footprint. These strategies may include optimizing server efficiency, utilizing renewable energy sources, and implementing energy-saving measures to reduce environmental harm and promote sustainability within the secure file storage system.

Safety

The security features of the project, such as access controls, encryption key management, and data integrity checks. Consider security best practices, including authentication and authorization mechanisms. Evaluate the safety and security aspects of the project, particularly with regard to data protection and confidentiality. Identify potential security threats and vulnerabilities, and outline strategies to mitigate them. Consider the legal and regulatory requirements for data security. The project's security features encompass robust access controls, encryption key management, and data integrity checks to ensure comprehensive protection. Following security best

practices, authentication and authorization mechanisms authenticate user identities and enforce access privileges. Safety and security considerations prioritize data protection and confidentiality, addressing potential threats like unauthorized access or data breaches. Mitigation strategies may include regular security audits, encryption protocols, and intrusion detection systems. Compliance with legal and regulatory requirements for data security, such as GDPR or HIPAA, further reinforces the project's commitment to safeguarding sensitive information, fostering trust among users and stakeholders.

Ethics

Examine the project's ethical consequences, taking into account concerns with access control, data security, and privacy. Take into account any potential effects on user data as well as any legal or legal requirements connected to user consent and data protection. It is essential to prioritize data privacy and security through robust encryption. Ethical considerations surrounding the project entail examining access control, data security, and privacy concerns. Prioritizing data privacy and security through robust encryption mechanisms is crucial, ensuring that user data remains protected from unauthorized access or breaches. Compliance with legal requirements, such as obtaining user consent and adhering to data protection regulations like GDPR or CCPA, is essential. Transparency regarding data usage and adherence to ethical guidelines foster trust between the system and its users, mitigating potential ethical dilemmas. By upholding principles of integrity and respect for user privacy, the project strives to navigate ethical complexities responsibly and ethically.

Cost

Estimation of Cost depends on various factors, including development, infrastructure, security, personnel, testing, compliance, maintenance, scaling, training, and incident response. You may need to invest in servers, storage devices, and networking equipment to set up a local host for file storage. The cost will depend on your storage requirements and redundancy needs. Estimating the project's cost entails considering various factors such as development, infrastructure, security, personnel, testing, compliance, maintenance, scaling, training, and incident response. Investments may be required for servers, storage devices, and networking equipment to establish a local host for file storage. Costs will vary based on storage requirements

and redundancy needs, with additional expenses incurred for security measures, personnel training, regulatory compliance, and ongoing maintenance. Scalability considerations should also be factored in to accommodate future growth and ensure the system's resilience. Proper budget allocation and cost analysis are essential for effectively managing expenses and optimizing resource utilization throughout the project lifecycle.

Type

The type of project "Secure File Storage using Hybrid Cryptography" falls under the category of Information Security and Cryptography. It aims to secure digital files through a combination of cryptographic techniques, making it a cybersecurity project. The encryption methods used fall under the broader category of cryptography, which is a fundamental aspect of cybersecurity . By leveraging cryptographic techniques, it safeguards digital files, underscoring the fundamental role of cryptography in cybersecurity endeavours. It fortifies digital file security, emphasizing the pivotal role of cryptography in modern cybersecurity initiatives aimed at protecting sensitive information from unauthorized access and breaches.

4 SYSTEM DESIGN

4.1 SYSTEM ANALYSIS

Designing a system for secure file storage on a local host using hybrid cryptography involves several components and considerations. Hybrid cryptography combines the strengths of symmetric and asymmetric encryption to provide both efficiency and security. Hybrid cryptography harnesses the strengths of both symmetric and asymmetric encryption techniques to achieve a balance between efficiency and security. Symmetric encryption ensures fast data processing, while asymmetric encryption enhances key management and distribution. By integrating these approaches, the system can offer robust protection against unauthorized access and data breaches, ensuring the confidentiality and integrity of stored files in the local host environment. Additionally, factors such as key generation, encryption algorithms, and secure storage mechanisms play pivotal roles in the design and implementation of such a system. Below is a high-level system design for such a system:

- **Requirements and Goals:** Define the specific security requirements, such as data confidentiality, integrity, and availability. Identify the types of files to be stored and their sensitivity. Determine the performance and usability requirements. The project's requirements and goals are crucial for defining its scope and objectives. Specific security requirements, including data confidentiality, integrity, and availability, must be clearly outlined to ensure comprehensive protection of stored files. Identification of file types and their sensitivity levels enables the implementation of appropriate security measures tailored to each category. Performance and usability requirements, such as system responsiveness and user interface intuitiveness, are essential for ensuring an efficient and user-friendly file storage solution. By delineating these requirements and goals, the project can effectively align its design and development efforts with the desired outcomes, ultimately delivering a secure and functional file storage system.
- **System Architecture:** The system architecture comprises several key components, each playing a vital role in ensuring secure file storage. The client

application serves as the user interface, facilitating file upload, download, and management operations. On the server side, a robust infrastructure hosts the file storage system, managing user requests and storing encrypted files securely. At the heart of the system lie cryptographic modules, responsible for encryption, decryption, and key management functions. These modules leverage hybrid cryptography techniques to safeguard data confidentiality and integrity. Together, these components form a cohesive system architecture that enables seamless and secure file storage on the local host.

- **User Authentication:** User authentication is a critical component of the system, ensuring that only authorized individuals can access sensitive data. A secure authentication mechanism will be implemented, utilizing various methods such as username/password, multi-factor authentication (MFA), or biometric authentication. Username/password authentication provides a baseline level of security, while MFA adds an extra layer of protection by requiring users to provide multiple forms of verification. Biometric authentication offers enhanced security by verifying users' unique physiological characteristics, such as fingerprints or facial recognition. By employing these authentication methods, the system reinforces access control measures and safeguards against unauthorized access, enhancing overall security and user trust. Implement a secure authentication mechanism to ensure that only authorized users can access the system. This can include username/password, multi-factor authentication, or biometric authentication.
- **File Upload and Storage:** In the proposed system, clients upload files to the server after encrypting them locally using a symmetric encryption algorithm, such as AES (Advanced Encryption Standard). This approach ensures that files remain secure during transmission and storage. Each file is encrypted with a unique symmetric key, enhancing security by preventing unauthorized access to multiple files if one key is compromised. Encrypting files on the client side before uploading them to the server minimizes the risk of interception or unauthorized access, bolstering data confidentiality and integrity within the file storage system. Clients upload files to the server. Encrypt files on the client

side before uploading them to the server using a symmetric encryption algorithm (e.g., AES). Each file should have a unique symmetric key.

- **Key Management:** Incorporating a hybrid cryptography approach, the system generates a unique pair of asymmetric keys for each user, comprising a public key for encryption and a private key for decryption. When a user uploads a file, the client application generates a random symmetric key specifically for that file. This symmetric key is then encrypted using the recipient's public key, ensuring that only the intended recipient possessing the corresponding private key can decrypt the symmetric key. Use a hybrid cryptography approach by generating a pair of asymmetric keys for each user. This consists of a public key (used for encryption) and a private key (used for decryption). When a user uploads a file, the client generates a random symmetric key for that file and encrypts it using the recipient's public key. Subsequently, the encrypted file, along with the encrypted symmetric key, is transmitted to the server and securely stored. This method leverages the efficiency of symmetric encryption for encrypting the file content while harnessing the robustness of asymmetric encryption for securely transmitting the symmetric key. The symmetric key is sent along with the encrypted file to the server, where it is securely stored. By employing this hybrid cryptography approach, the system enhances security by combining the advantages of both symmetric and asymmetric encryption techniques. It ensures confidentiality and integrity during file transmission and storage, mitigating the risk of unauthorized access or data breaches within the file storage system.
- **File Encryption and Decryption:** When a user wants to access their file, the server sends the encrypted file along with the encrypted symmetric key. The client decrypts the symmetric key using the user's private key. Using the decrypted symmetric key, the client decrypts the file. Firstly, the client decrypts the symmetric key using the user's private key, which is securely stored and accessible only to the user. This decryption process ensures that the symmetric key is accessible only to the intended recipient, providing a crucial layer of security. Once the symmetric key is successfully decrypted, the client utilizes this key to decrypt the encrypted file content. Decrypting the file with

the symmetric key restores the original plaintext data, allowing the user to access and view their file securely. This decryption process occurs entirely on the client side, maintaining the confidentiality of the user's private key and ensuring that sensitive data remains protected throughout the file retrieval process. By employing this approach, the system upholds data confidentiality and integrity while ensuring secure file access for authorized users. Additionally, the use of asymmetric encryption for transmitting the symmetric key enhances security by limiting access to the symmetric key to only the intended recipient, thereby mitigating the risk of unauthorized access to user files.

- **Access Control:** Implement access control mechanisms to restrict file access to authorized users. Ensure that users can only access files they are authorized to view or modify. Access control mechanisms are essential for ensuring that only authorized users can access and manipulate files within the system. Implementing robust access control measures involves defining and enforcing permissions based on user roles, privileges, or other criteria. Users should only be granted access to files they are authorized to view or modify, thereby safeguarding data confidentiality and integrity. This can be achieved through role-based access control (RBAC), attribute-based access control (ABAC), or other access control models tailored to the system's requirements. By implementing effective access control mechanisms, the system mitigates the risk of unauthorized access and maintains data security and privacy.

4.1.1 System Architecture

An architectural diagram in software engineering is a visual representation of the high-level structure and components of a software system. It serves as a blueprint for understanding how various parts of a software application or system interact with each other and how data flows between them.

These diagrams help software engineers, developers, and stakeholders to visualize and communicate the system's design, making it easier to discuss, plan, and implement software solutions.

Moreover, architectural diagrams serve as a common language for communication among team members, fostering collaboration and alignment on design decisions. They empower stakeholders to visualize the system's design, aiding in the identification of potential bottlenecks, dependencies, and scalability concerns.

Through clear visualization of data flows and component interactions, these diagrams facilitate more informed discussions, enabling teams to iteratively refine and improve the software solution.

The diagram below is a simple block diagram explaining the basic working of the overall projects covering three main steps – file upload, cryptography and file download. We make use of both AES and RSA algorithms which are symmetric and asymmetric respectively, collectively making it a hybrid method of cryptography.

Using the AES algorithm helps enhance the speed of file upload and file download while the RSA algorithm takes care of the security of the file.

In other words, the symmetric key generated by the AES algorithm is further protected by the key generated in RSA algorithm. Moreover, architectural diagrams serve as a universal language for team communication, fostering collaboration and alignment on design decisions. They enable stakeholders to visualize the system's design, identify potential bottlenecks, dependencies, and scalability concerns.

By illustrating data flows and component interactions clearly, these diagrams facilitate informed discussions, allowing teams to iteratively refine and improve the software solution. In the depicted diagram, a simple block diagram outlines the key steps of the project: file upload, cryptography, and file download.

The integration of AES and RSA algorithms represents a hybrid cryptography approach, blending symmetric and asymmetric encryption for enhanced security and efficiency.

AES optimizes file upload and download speeds, while RSA ensures file security by safeguarding the symmetric key generated by AES.

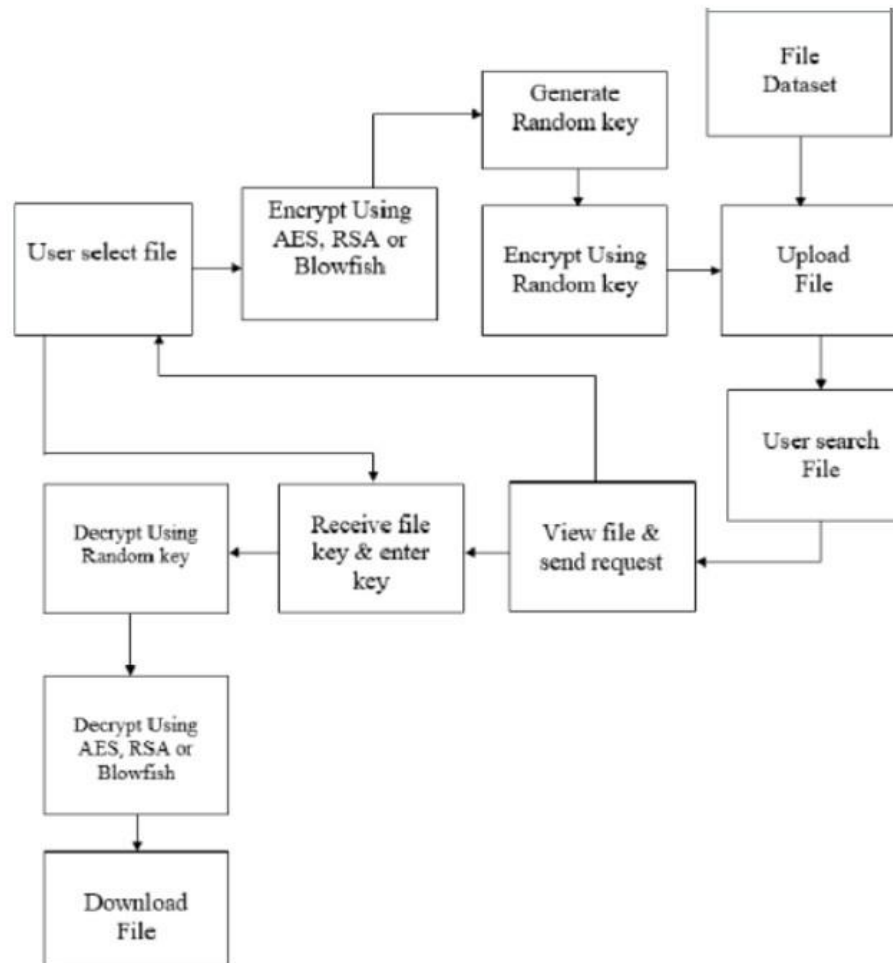


Fig 4.1.1 Block Diagram

4.2 UML Diagrams

Unified Modelling Language (UML), as a standardized modelling language, serves as the lingua franca of software engineering, facilitating seamless communication and collaboration among diverse stakeholders. Beyond its application in software development, UML finds utility in diverse fields, including business process modelling, system analysis, and even in the realm of project management.

By providing a comprehensive framework for visualizing, designing, specifying, and documenting software systems and processes, UML empowers teams to navigate the complexities of system architecture with clarity and precision.

UML diagrams, ranging from structural diagrams like class diagrams and component diagrams to behavioural diagrams like sequence diagrams and state diagrams, offer a rich visual vocabulary for representing various aspects of system behaviour and

structure. These diagrams not only aid in conceptualizing and articulating system requirements but also serve as invaluable artifacts throughout the software development lifecycle.

They provide a common reference point for discussing design decisions, identifying potential risks and dependencies, and validating system functionality against stakeholder expectations.

As such, UML emerges as an indispensable tool for fostering collaboration, driving innovation, and ultimately delivering software solutions that meet the evolving needs of stakeholders in an ever-changing technological landscape.

4.2.1 Sequential Diagram

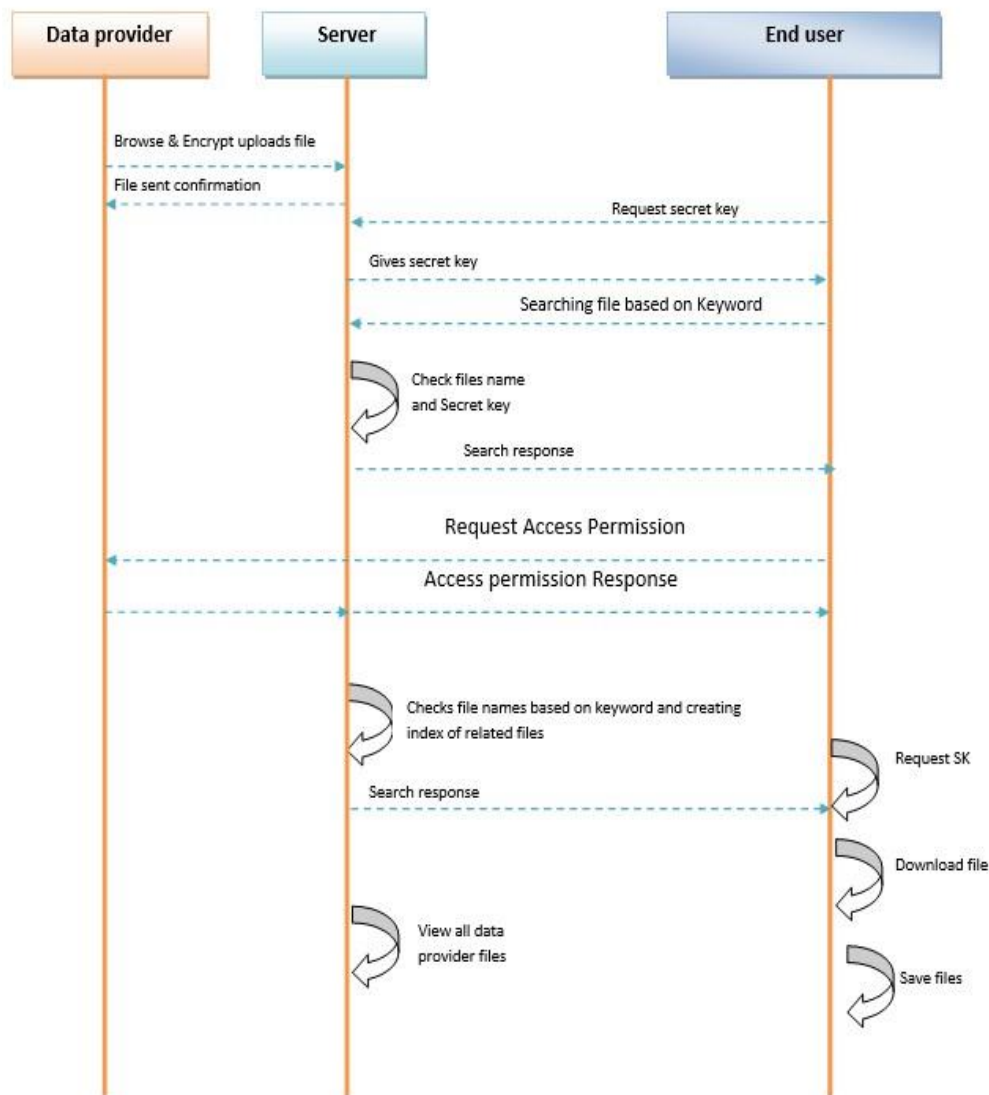


Fig 4.2.1 Sequential Diagram

4.2.2 Activity Diagram

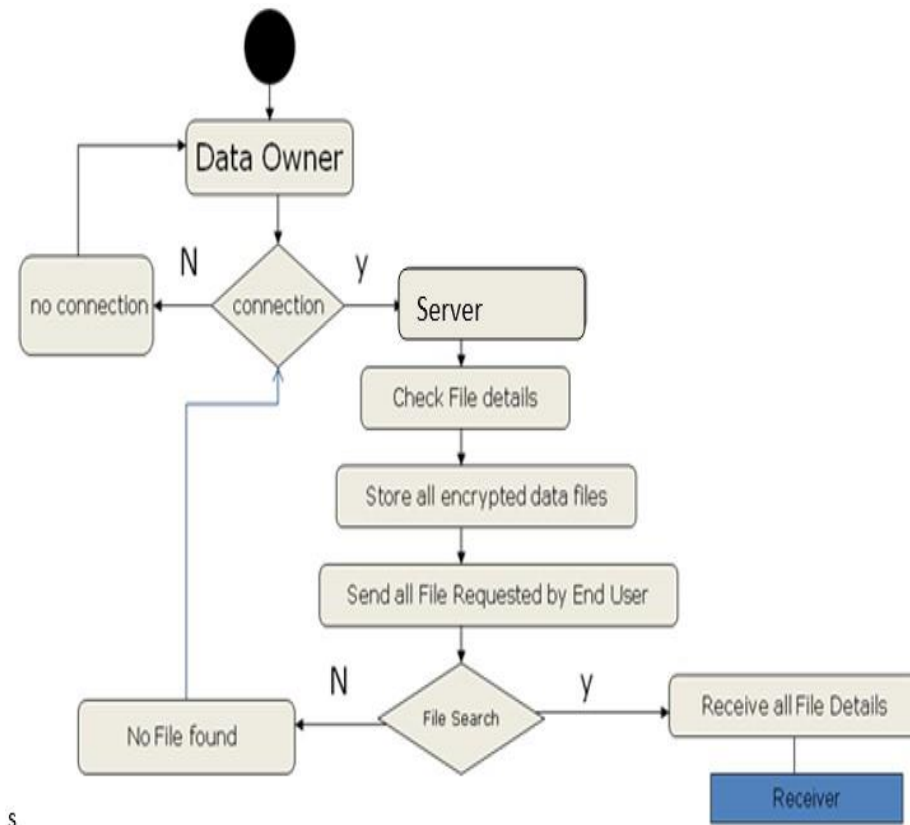


Fig 4.2.2 Activity Diagram

4.2.3 Use case diagram

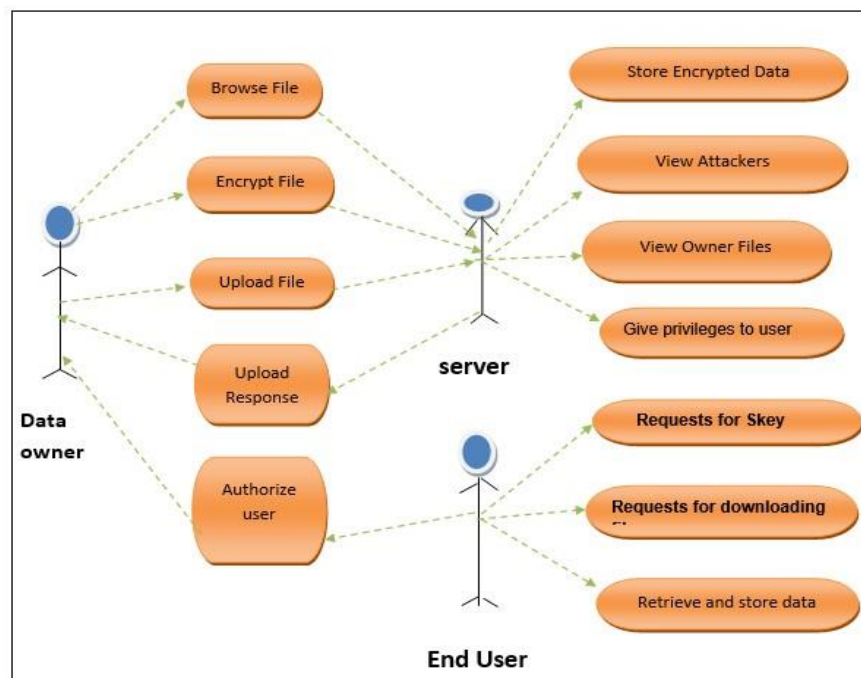


Fig 4.2.3 Use Case Diagram

In conclusion, UML diagrams are a powerful tool for understanding, designing, and communicating complex systems. They play a vital role in software development and find applications in various domains beyond software engineering. By utilizing UML diagrams, stakeholders can ensure that their systems are well-designed, maintainable, and fulfil their intended purpose.

4.2.4 Class Diagram

Class diagrams represent the classes in a system and the associations between them. They are fundamental for object-oriented design and show the blueprint of the system's structure. Class diagrams are essential tools in object-oriented design, providing a visual representation of the classes within a system and the relationships between them. Each class in the diagram represents a blueprint for objects of that class, encapsulating both data (attributes) and behaviours (methods). Associations between classes illustrate how objects interact and collaborate to achieve system functionality. In a class diagram, classes are depicted as boxes with three compartments: the top compartment contains the class name, the middle compartment lists class attributes, and the bottom compartment shows class methods. Associations between classes are represented by lines connecting the related classes, often annotated with multiplicities to indicate the cardinality of the association. Additionally, class diagrams can include various types of associations, such as aggregation, composition, inheritance, and dependency, to depict more complex relationships between classes. Aggregation and composition represent whole-part relationships, inheritance illustrates the "is-a" relationship between classes, and dependency denotes a weaker form of association where one class relies on another class for functionality. Overall, class diagrams serve as blueprints for designing object-oriented systems, aiding in the visualization, organization, and communication of system structures and relationships. They are invaluable tools for software developers, architects, and stakeholders in understanding and refining the design of complex software systems.

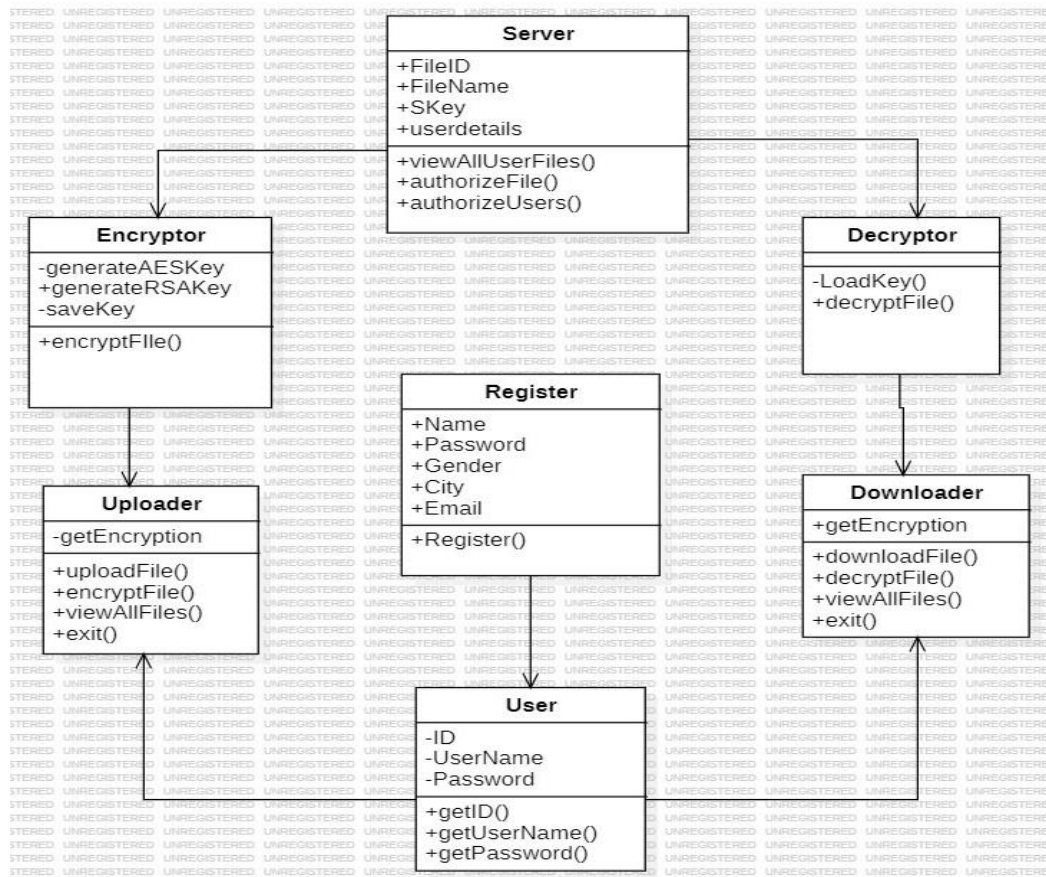


Fig 4.2.4 Class Diagram

4.3 SYSTEM dESIGN

4.3.1 Modular Design

1. User Authentication Module

This module is Responsible for authenticating users before granting access to the system. May use techniques such as password-based authentication on multi-factor authentication. Ensures only authorized users can interact with the file storage system.

2. File Upload and Download Module

Allows users to upload files securely to the system. Provides options for downloading and accessing stored files. May include features like file versioning and metadata storage.

3. Hybrid Cryptography Module

Combines symmetric and asymmetric encryption for secure file storage. Generates and manages encryption keys. Encrypts files using symmetric keys, which are in turn encrypted with the recipient's public key. Decrypts files using the recipient's private key and the stored symmetric key.

4. Key Management Module

Manage the creation, distribution, and storage of encryption keys. Ensure keys are securely stored and accessible only to authorized users. Provides key rotation and revocation mechanisms for enhanced security.

5. Access Control Module

Defines and enforces access control policies for files. Allows users to specify who can access, modify, or delete their files. May integrate role-based access control (RBAC) for more complex permission structures. The Access Control module is critical for maintaining the security and integrity of the file storage system, defining and enforcing policies to regulate user access to files. This module empowers users to specify permissions for their files, dictating who can access, modify, or delete them. By implementing granular access control mechanisms, the system ensures that only authorized individuals can perform specific actions on files, mitigating the risk of unauthorized access or malicious activities. Moreover, the module may integrate role-based access control (RBAC) to facilitate more complex permission structures. RBAC assigns roles to users based on their responsibilities or job functions, allowing for streamlined access management across the organization. Through RBAC, permissions can be assigned to roles rather than individual users, simplifying administration and enhancing scalability. Overall, the Access Control module plays a pivotal role in safeguarding data confidentiality and integrity, providing users with the necessary controls to manage access to their files effectively while adhering to security best practices.

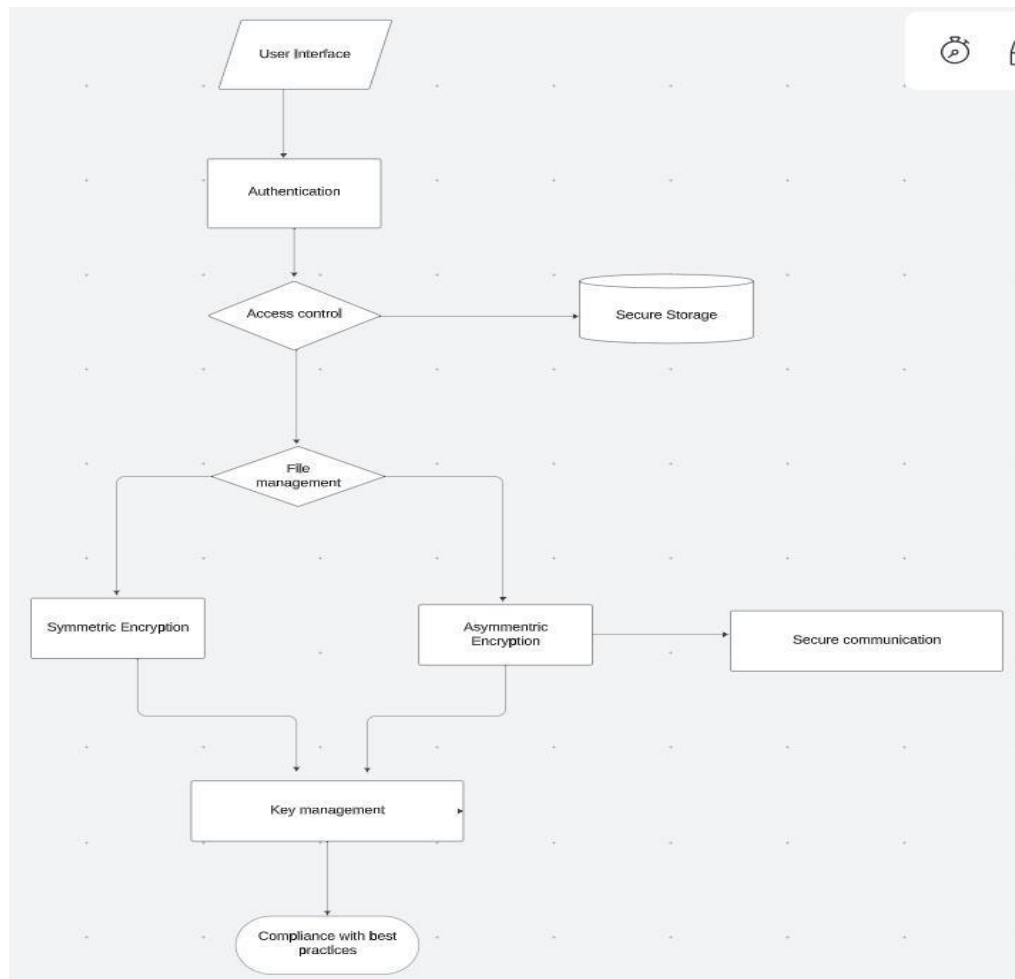


Fig 4.3.1 Dataflow diagram

4.3.2 Database Design

Database Initialization

The SQL dump begins by setting some SQL modes and time zone settings. It then creates a database named `cyber_insurance` if it doesn't already exist using the `CREATE DATABASE` statement.

Table Creation

The dump contains a series of `CREATE TABLE` statements that define the structure of various database tables. Each `CREATE TABLE` statement specifies the table's name, the columns it contains, data types for each column, constraints, and indexes.

Table Data Insertion

After creating the table structure, the dump includes INSERT INTO statements that populate some of the tables with initial data. These statements specify the columns and values to be inserted.

Constraints and Relationships

The SQL dump also includes ALTER TABLE statements that define constraints and relationships between tables. These statements establish foreign key relationships between tables, ensuring data integrity and referential integrity in the database.

Table Constraints

Constraints, such as primary keys, unique keys, and indexes, are defined for each table to enforce data integrity rules.

Database Version Control

The dump may include a table called Django migrations, which is used by Django (a web framework) to track and manage database schema changes over time. It records which database migrations have been applied.

Algorithm Used- Advanced Encryption Algorithm [AES] - The Advanced Encryption Standard (AES) algorithm plays a crucial role in the project "Secure File Storage on Local Host using Hybrid Cryptography" by providing strong encryption for the actual file data stored on the local host.

1. Key Generation

To begin the encryption process, a symmetric encryption key is generated. This key is required for both the encryption and decryption of files.

The symmetric key can be generated locally, or it can be provided by the user. The choice of generating the key locally or letting the user input it depends on the project's design and security policies.

Ensuring robust encryption, a symmetric key is generated at the outset, indispensable for both encryption and decryption processes. The choice between local generation or user-provided keys is pivotal, aligning with project intricacies and stringent security protocols. This decision underscores the project's commitment to tailored data protection strategies, where each approach is meticulously considered to fortify the integrity of encryption mechanisms and safeguard sensitive information against potential threats and vulnerabilities.

2. File Encryption

When a user uploads a file to be stored securely, the AES algorithm is employed to encrypt the content of the file.

The symmetric encryption key generated or provided is used in this step. AES is a symmetric encryption algorithm, which means the same key is used for both encryption and decryption. AES breaks down the file into blocks of data, typically 128, 192, or 256 bits, depending on the chosen AES key length. Each block is then encrypted separately.

When a user uploads a file for secure storage, the AES algorithm encrypts its content. Utilizing the symmetric encryption key, AES encrypts the file in blocks of data,

typically 128, 192, or 256 bits, based on the chosen AES key length, enhancing data security through meticulous encryption.

3. Security of the Symmetric Key

While the symmetric key is necessary for encrypting and decrypting files, it must be securely managed. In the context of the hybrid cryptography model, the symmetric key is often encrypted using an asymmetric encryption algorithm, such as RSA. While essential for encryption and decryption, the symmetric key requires secure management. In a hybrid cryptography model, it's commonly encrypted using asymmetric algorithms like RSA. This approach enhances security by leveraging the strengths of both symmetric and asymmetric encryption techniques for robust data protection. This dual-layered encryption strategy ensures that even if one layer is compromised, the data remains safeguarded, bolstering overall security measures.

4. Symmetric Key Transmission or Storage

The encrypted symmetric key can be securely transmitted or stored. This is usually done using the recipient's public key, ensuring that only the intended recipient, who possesses the corresponding private key, can access the symmetric key. The encrypted symmetric key is securely transmitted or stored, often utilizing the recipient's public key. This method guarantees that solely the intended recipient, possessing the corresponding private key, can access the symmetric key, reinforcing confidentiality and thwarting unauthorized access attempts for enhanced data protection and security.

5. Data Storage

Encrypted files and the encrypted symmetric key are stored on the local host, ensuring that even if unauthorized access occurs, the data remains protected. Encrypted files and the symmetric key are securely stored on the local host, bolstering data protection measures. This ensures that even in the event of unauthorized access attempts, the encryption safeguards the data, maintaining its confidentiality and integrity.

6. File Retrieval and Decryption

When a user requests to download an encrypted file, the symmetric key is retrieved from secure storage or decrypted using the recipient's private key (in the case of transmission). The AES algorithm is then employed to decrypt the file's content using the retrieved symmetric key.

Upon a user's request to download an encrypted file, the symmetric key is fetched from secure storage or decrypted utilizing the recipient's private key, particularly in transmission scenarios. Subsequently, the AES algorithm is invoked to decrypt the file's content, leveraging the retrieved symmetric key.

This meticulous decryption process ensures seamless retrieval of the original file content, maintaining data integrity and confidentiality throughout the transmission and decryption stages, reinforcing overall security measures.

The use of AES in this project provides a robust and efficient means of encrypting and decrypting files. AES is known for its speed and security, making it an ideal choice for securing data in a file storage system.

The integration of AES and RSA with hybrid cryptography ensures that data is protected during both storage and transmission, making the project highly secure and reliable.

Working of AES

AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

128 bit key - 10 rounds

192 bit key - 12 rounds

256 bit key - 14 rounds

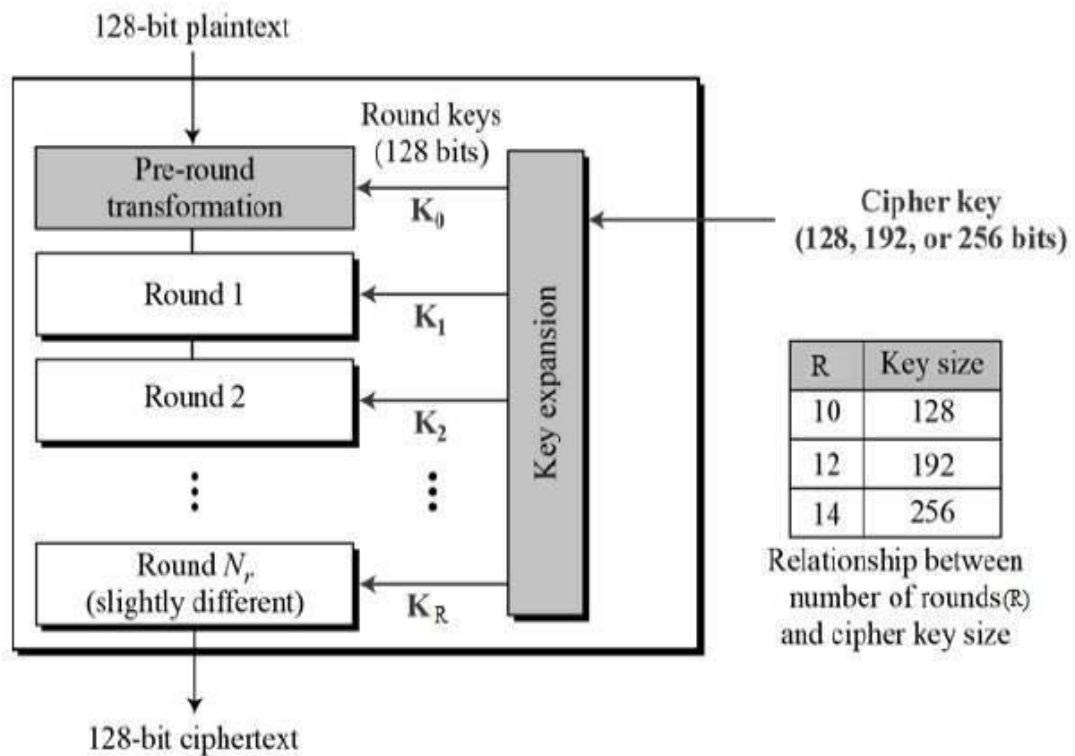


Fig 5.1.1 Working of AES Algorithm

Working of RSA Algorithm

RSA (Rivest-Shamir-Adleman) is a widely used asymmetric encryption algorithm that relies on the mathematical properties of prime numbers. Its working principle involves three main steps: key generation, encryption, and decryption.

RSA encryption is based on the difficulty of factoring large composite numbers into their prime factors. The algorithm relies on the mathematical properties of modular exponentiation and the Euler's totient function. The RSA algorithm is secure because of the difficulty of finding the prime factors of a large composite number, which forms the basis of its encryption and decryption processes. The security of RSA relies on the practical impossibility of factoring the product of two large prime numbers into its prime factors. As such, RSA encryption provides a high level of security against brute-force attacks, even with modern computing power.

The security of RSA relies on the difficulty of factoring the modulus n into its prime factors, making it computationally infeasible to derive the private key d from the public key (e, n) without knowing the prime factors of n . Despite its strengths, RSA does have limitations, particularly in terms of key size and performance. However, RSA remains a cornerstone of modern cryptography and continues to be extensively utilized in various applications.

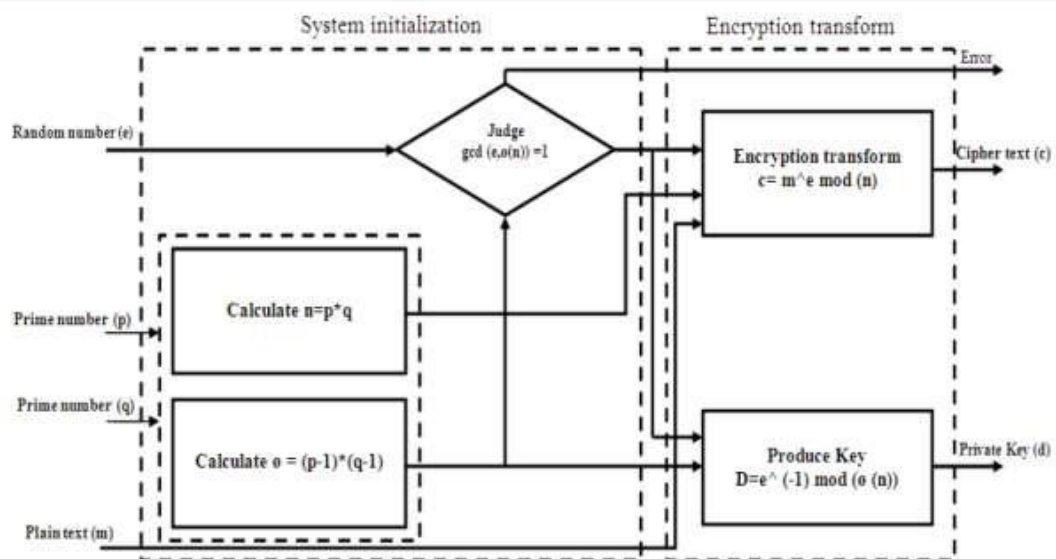


Fig 5.1.2 Working of RSA Algorithm

Creation of Round keys

A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption. In AES encryption, a Key Schedule algorithm computes all round keys from the initial key. This process generates multiple round keys, each tailored for a specific encryption round. By diversifying keys for each round, AES enhances security and resilience, fortifying data protection against cryptographic attacks and ensuring robust encryption. Similarly, in RSA encryption, a Key Generation algorithm generates the public and private key pair from user-defined parameters. These keys serve distinct purposes: the public key is utilized for encryption, while the private key is employed for decryption. This dual-key mechanism ensures secure communication and confidentiality in RSA encryption, enhancing overall data protection. Key Schedule algorithms in encryption, like those used in AES and RSA, play a crucial role in generating round keys or key pairs from the initial key. In AES, this diversification enhances security by creating unique round keys for each encryption round, while in RSA, it ensures secure communication through the generation of public and private key pairs for encryption and decryption.

5.2 SAMPLE CODE

models.py

```
class RegisterModel(models.Model):

    firstname=models.CharField(max_length=300)
    lastname=models.CharField(max_length=200)
    userid=models.CharField(max_length=200)
    password=models.IntegerField()
    mblenum=models.BigIntegerField()
    email=models.EmailField(max_length=400)
    gender=models.CharField(max_length=200)


class UploadModel(models.Model):

    file_name=models.CharField(max_length=100)
    category=models.CharField(max_length=100)
    upload_user=models.ForeignKey(RegisterModel)

    upload_file=models.FileField()

    area=models.CharField(max_length=200)
    add_count=models.IntegerField(default='0')


class RequestModel(models.Model):

    accessone=models.ForeignKey(RegisterModel)
    accesstwo =models.ForeignKey(UploadModel)
    cate=models.CharField(max_length=500)


class FeedbackModel(models.Model):

    username = models.ForeignKey(RegisterModel)

    feedback = models.CharField(max_length=300)
```

views.py

```
from random import randintfrom

from django.core.mail import EmailMultiAlternatives

from django.shortcuts import render, redirect, get_object_or_404


# Create your views here.
```

```

from Designing_Cyber_Insurance_Policies.settings import
DEFAULT_FROM_EMAIL

from user.forms import RegisterForms

from user.models import RegisterModel, UploadModel, RequestModel,
FeedbackModel

def index(request):
    usid=""

    if request.method=="POST":
        usid=request.POST.get('username')
        pswd = request.POST.get('password')

        try:

            check=RegisterModel.objects.get(user
            id=usid,password=pswd)
            request.session['userid']=check.id
            return redirect('user_page')

        except: pass

        return render(request,'user/index.html',{'fh':usid})


def register(request):

    if request.method=="POST":
        forms=RegisterForms(request.POST)
        if forms.is_valid():

            forms.save()
            return redirect('index')

        else:

            forms=RegisterForms()
            return
            render(request,'user/register.html',{'form':forms})


def user_page(request):

    uid = request.session['userid']

    request_obj = RegisterModel.objects.get(id=uid)

    myfile=""

```

```

a=""

b=""

c=""

d=""

if request.method=="POST" and request.FILES['myfile']:

    myfile = request.FILES['myfile']
    a=request.POST.get('name')

    b=request.POST.get('category')

    c=request.POST.get('area')


    UploadModel.objects.create(file_name=a,category=b,upload_user=request_obj,upload
    _file=myfile,area=c)

    return render(request,'user/user_page.html')


def upload_fileview(request):

    uid=""

    sts
    ='pending'
    sent = 'sent'

    uid = request.session['userid']

    request_obj = RegisterModel.objects.get(id=uid)
    obj=UploadModel.objects.filter(upload_user=request_obj
    j)    if request.method=="POST":
        uid=request.session['userid']


        request_obj=RegisterModel.objects.get(id=uid)

        subject = "OTP"
        text_content = ""

        otp = randint(1000, 9999)

        request.session['otp']=otp

        html_content = "<br/><p>OTP :<strong>" + str(otp) + "</strong>/p>"

```

```

        from_mail = DEFAULT_FROM_EMAIL

        to_mail = [request_obj.email]

        if send_mail(subject,message,from_mail,to_mail):
            msg = EmailMultiAlternatives(subject, text_content, from_mail, to_mail)

            msg.attach_alternative(html_content, "text/html")

        if msg.send():
            sts='sent'

            #return redirect('otppage',lower=userObj.id)

        return render(request,'user/upload_fileview.html',{'obj':obj,'sts':sts,'sent':sent,})

def      otplib(request,pk):
password      =
request.session['otp']

sts = "c"

pas
=type(password)

ss=""

count=0

aaa=""
vott,vott1=0,
0

pkid=UploadModel.objects.get(id=pk)

aaa=pkid.id
request.session['jhf']=aaa
if request.method ==
"POST":

objs = UploadModel.objects.get(id=pk)

unid = objs.id

vot_count = UploadModel.objects.all().filter(id=unid)
for t in vot_count:

vott = t.add_count

vott1 = vott + 1

obj = get_object_or_404(UploadModel, id=unid)

```



```

obj.add_count = vott1

obj.save(update_fields=["add_count"])

onetime = request.POST.get('otp', "")
ss = onetime

if int(password) == int(onetime):

    return redirect('download_page')
else:

    sts = "Please Enter Correct OTP"

    return render(request, 'user/otppage.html', {'password': pas, 'sts': sts, 'count': aaa})

def download_page(request):
    count=0

    aaaa= request.session['jhf']
    obj=UploadModel.objects.filter(id=aaaa)

    if request.method == "POST":

        obj = get_object_or_404(UploadModel, id=obj)

        obj.add_count = count

        obj.save(update_fields=["add_count"])

        return redirect('upload_fileview')
    return render(request, 'user/download_page.html', {'a': aaaa, 'obj': obj})
def request(request, pk):

    userid = request.session['userid']

    uobj =
    RegisterModel.objects.get(id=userid)
    obj = UploadModel.objects.get(id=pk)

    a=obj.category

    RequestModel.objects.create(accessone=uobj, accesstwo=obj, cate=a)
    return redirect('download_page')

def view_file(request):

    uid = request.session['userid']

    request_obj = RegisterModel.objects.get(id=uid)

```

```

obj =
UploadModel.objects.filter(upload_user=request_obj)
return render(request,'user/view_file.html',{'obj':obj})

def send_feedback(request):

uid = request.session['userid']

objec =
RegisterModel.objects.get(id=uid)    if
request.method == "POST":

    feed = request.POST.get('feedback')

    FeedbackModel.objects.create(username=object,    feedback=feed)
return render(request,'user/send_feedback.html')

def mydetails(request):

    usid = request.session['userid']

us_id =
RegisterModel.objects.get(id=usid)
return
render(request,'user/mydetails.html',{'obje
':us_id})

```

emailsettings.py

```

SET_EMAIL_USE_TLS = True

SET_EMAIL_HOST='smtp.gmail.com'

SET_EMAIL_HOST_USER='chandanareddy0123@gmail.com'

#dskreddy02@gmail.com'

SET_EMAIL_HOST_PASSWORD='*****'

SET_EMAIL_PORT=587

SET_EMAIL_BACKEND=

'django.core.mail.backends.smtp.EmailBackend'

SET_DEFAULT_FROM_EMAIL = 'chandanareddy0123@gmail.com'

```

6 TESTING

6.1 SOFTWARE TESTING

Software testing is a critical process in the development lifecycle that ensures the quality, functionality, and reliability of software products. It involves systematically executing a software application or system to identify defects, errors, or bugs, and to verify that it meets specified requirements and performs as expected.

The importance of software testing cannot be overstated. Firstly, it helps identify and rectify defects early in the development cycle, minimizing the cost and effort required for fixes later on. By detecting and addressing issues before deployment, testing enhances the overall quality of the software, reducing the likelihood of post-release failures or malfunctions.

Moreover, software testing provides assurance to stakeholders, including developers, clients, and end-users, that the software meets their expectations and performs reliably under various conditions. This builds trust and confidence in the product, leading to increased user satisfaction and loyalty.

Additionally, testing helps mitigate risks associated with software failures, such as financial losses, damage to reputation, or compromised security. By uncovering vulnerabilities and weaknesses, testing enables developers to implement necessary security measures and ensure compliance with industry standards and regulations.

In essence, software testing plays a pivotal role in delivering high-quality, robust, and reliable software solutions that meet user needs, enhance productivity, and drive business success in today's competitive marketplace.

6.1.1 Unit Testing

Unit testing is a fundamental aspect of software development aimed at verifying the correctness of individual units or components of a software application. In unit testing, each unit, typically a function or method, is isolated and tested independently to ensure it behaves as expected. This approach allows developers to identify bugs early in the development process, as units are tested in isolation from the rest of the system. Unit tests are written by developers and are usually automated,

meaning they can be run repeatedly with minimal effort. Test cases are created to cover various scenarios, including typical inputs, edge cases, and error conditions. By running these tests regularly, developers can catch regressions introduced by code changes and ensure that existing functionality remains intact. It promotes code modularity, as units are designed to be testable in isolation. It also encourages better code design by forcing developers to consider the interface and dependencies of each unit. Overall, unit testing plays a crucial role in maintaining code quality, reducing bugs, and increasing confidence in the software's correctness and reliability.

6.1.2 Integration Testing

Integration testing is a phase of software testing where individual units or components of a system are combined and tested as a group. The primary goal is to ensure that these integrated components work together seamlessly and function correctly as a whole. Unlike unit testing, which focuses on testing individual units in isolation, integration testing examines interactions between units, as well as their interfaces and dependencies. Integration tests verify that data flows correctly between components, that communication between modules is reliable, and that the system behaves as expected when different parts are integrated. There are several approaches to integration testing, including top-down, bottom-up, and sandwich testing. Top-down testing starts with the highest-level modules and gradually integrates lower-level modules, while bottom-up testing begins with the lowest-level modules and works upwards. Sandwich testing combines elements of both approaches by testing modules at various levels simultaneously. Integration testing helps identify issues such as interface mismatches, data flow problems, and integration errors, ensuring the overall stability and functionality of the software system.

6.1.3 User Acceptance Testing

User Acceptance Testing (UAT) is the final phase of testing before a software product is released to end-users. Unlike integration testing, which focuses on technical aspects, UAT evaluates the software's readiness for real-world use from the user's perspective. In UAT, end-users or their representatives execute predefined test cases to validate whether the software meets their requirements and expectations. UAT aims

to ensure that the software aligns with user needs, is intuitive to use, and fulfils its intended purpose effectively. Test cases typically cover common user scenarios, workflows, and business processes to simulate real-world usage scenarios.

UAT provides valuable feedback to stakeholders, allowing them to identify any discrepancies between the software and user expectations. It also validates that any issues raised during earlier testing phases have been addressed satisfactorily.

Successful completion of UAT signals that the software is ready for deployment, instilling confidence in both users and stakeholders that the product will meet their needs and deliver the expected value.

6.1.4 Validation Checking

Validation checks play a crucial role in ensuring the integrity and reliability of data input in software systems. In the context of the mentioned fields, such as text and numeric inputs, validation ensures the only appropriate data is accepted, minimizing the risk of errors and ensuring data accuracy.

For text fields, the validation encompasses size constraints and alphanumeric or alphabetic character requirements, triggering error messages for incorrect entries. Numeric fields, on the other hand, are restricted to numeric characters only, with any deviation prompting error notifications.

- **Text Field:** The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes an error message.
- **Numeric Field:** The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to a test run along with sample data.

Following individual module testing, integration into a cohesive system occurs, allowing for comprehensive testing with real data. This process aims to detect any program defects by analysing the system's output against expected results.

Effective testing planning ensures that all system requirements are thoroughly examined, with successful tests identifying defects related to inappropriate data input

and system errors. Ultimately, validation checks and testing procedures are essential components of software development, contributing to the creation of robust and reliable system.

The individually tested modules are integrated into a single system. Testing involves executing the real data information used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces an output revealing the errors in the system.

6.1.5 Output Testing

Output testing is a critical aspect of software testing that focuses on verifying the correctness and accuracy of the output generated by a software application. It ensures that the system produces the expected results based on given inputs and requirements. Output testing encompasses various types of testing techniques and strategies to validate different aspects of the system's output, including functional correctness, data integrity, formatting, and performance. Functional correctness testing involves verifying that the output produced by the software matches the expected output defined in the system requirements or specifications. Test cases are designed to cover a range of scenarios to ensure that the software behaves correctly under different conditions. This includes testing boundary conditions, edge cases, and error handling scenarios to validate the robustness of the output.

Data integrity testing focuses on ensuring the accuracy and consistency of the data presented in the output. It involves comparing the output data against known input data or expected results to detect any discrepancies or inconsistencies. This type of testing is crucial for applications that manipulate or process large volumes of data to ensure data accuracy and reliability.

Formatting testing ensures that the output is presented in the desired format, layout, and style according to user expectations and design specifications. This includes testing for proper alignment, font styles, colours, and other visual elements to ensure a consistent and professional appearance. Performance

testing evaluates the responsiveness and efficiency of the software in generating output within acceptable time frames. This includes testing for response times, throughput, and resource utilization to identify any performance bottlenecks or scalability issues that may affect the system's output quality and usability.

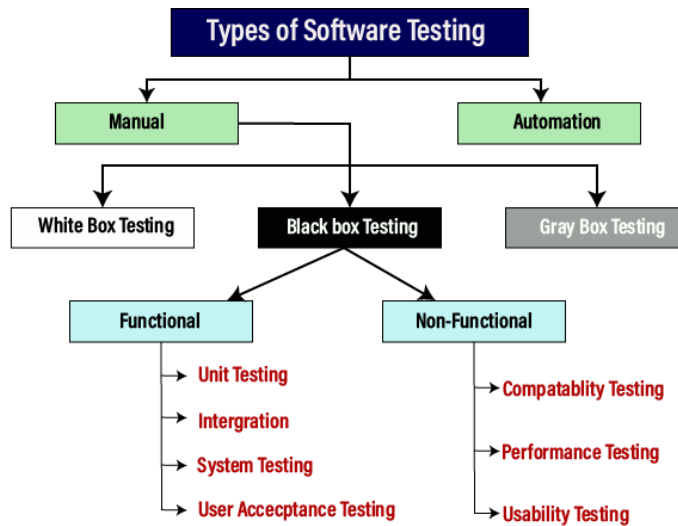


Fig 6.1.5 Types of Software Testing

6.2 PLAGIARISM REPORT

A plagiarism report is a document generated by specialized software that identifies similarities between a submitted text and existing sources. It highlights matched text passages and provides details on their origins, such as web pages or publications.

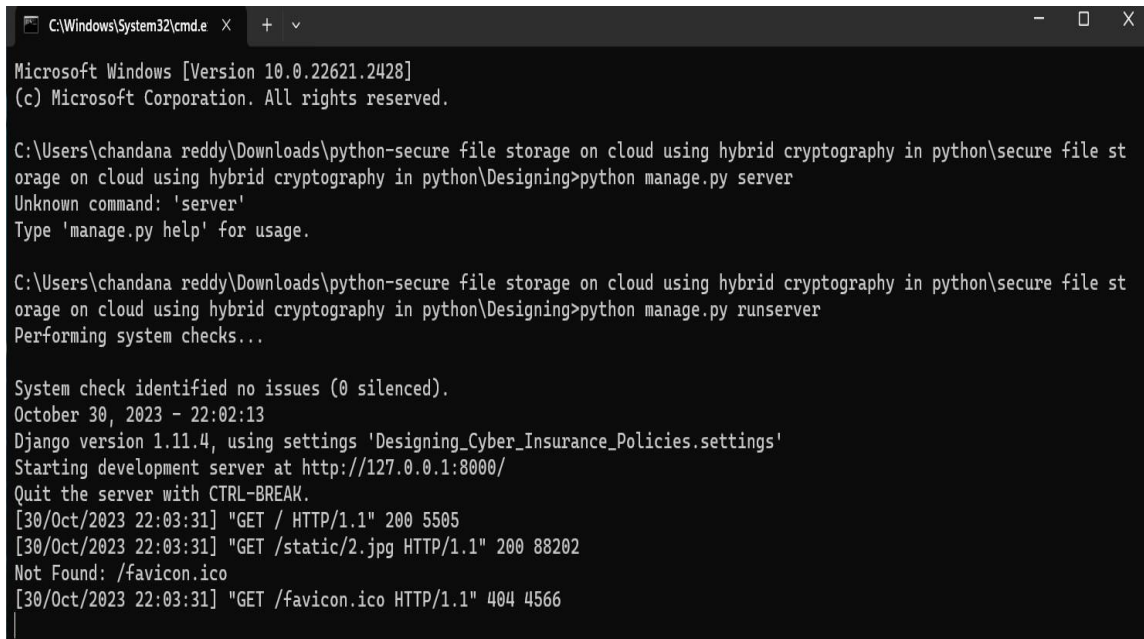


Fig 6.2 Plagiarism Report

6.3 TEST CASES

| Test Id | Test Name | Input | Output | Expected Result | Status |
|---------|--|--|---|-------------------------------|--------|
| 1 | Owner browsing the data | file | Data stored | Data stored in encrypted form | PASS |
| | Owner browsing data | data | Data is not Present in directory | Data stored in encrypted form | FAIL |
| 2 | Owner Verification of Files | File Name with Uploaded MAC In Cloud and Auditor | File Attacked or Safe | Attacked or Safe | PASS |
| | Owner Verification of Files | No Inputs | No Mac to Compare File | Attacked or Safe | FAIL |
| 3 | User Access File MAC & SK | File Name | Get MAC and SK if Cloud Permitted | Get SK and MAC | PASS |
| | User Access File MAC & SK | File Name | Should not Get MAC and SK if Cloud not Permitted | Get SK and MAC | FAIL |
| 4 | User Downloads file content | File Name | Get decrypted content if Cloud Permitted | Download Content | PASS |
| | User Downloads file content | File Name | Should not Get decrypted content if Cloud not Permitted | Download Content | FAIL |
| 5 | End user downloading data from server | Server IP address | Downloaded Data from the server | Data should be downloaded | PASS |
| | End user downloading data from server | Invalid server IP address | Cant Downloaded Data from the server | Data should be downloaded | FAIL |
| 6 | Attacker (Hacking) | File Name | Get File Details | Hack the content | PASS |
| | Attacker(Hacking) With Same credential of Attacker | File Name | Get File Details | Hack the content | Fail |

Table 6.3 Test Cases Report



```

C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22621.2428]
(c) Microsoft Corporation. All rights reserved.

C:\Users\chandana reddy\Downloads\python-secure file storage on cloud using hybrid cryptography in python\secure file st
orage on cloud using hybrid cryptography in python\Designing>python manage.py server
Unknown command: 'server'
Type 'manage.py help' for usage.

C:\Users\chandana reddy\Downloads\python-secure file storage on cloud using hybrid cryptography in python\secure file st
orage on cloud using hybrid cryptography in python\Designing>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
October 30, 2023 - 22:02:13
Django version 1.11.4, using settings 'Designing_Cyber_Insurance_Policies.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[30/Oct/2023 22:03:31] "GET / HTTP/1.1" 200 5505
[30/Oct/2023 22:03:31] "GET /static/2.jpg HTTP/1.1" 200 88202
Not Found: /favicon.ico
[30/Oct/2023 22:03:31] "GET /favicon.ico HTTP/1.1" 404 4566

```

Fig 7.1 Running server

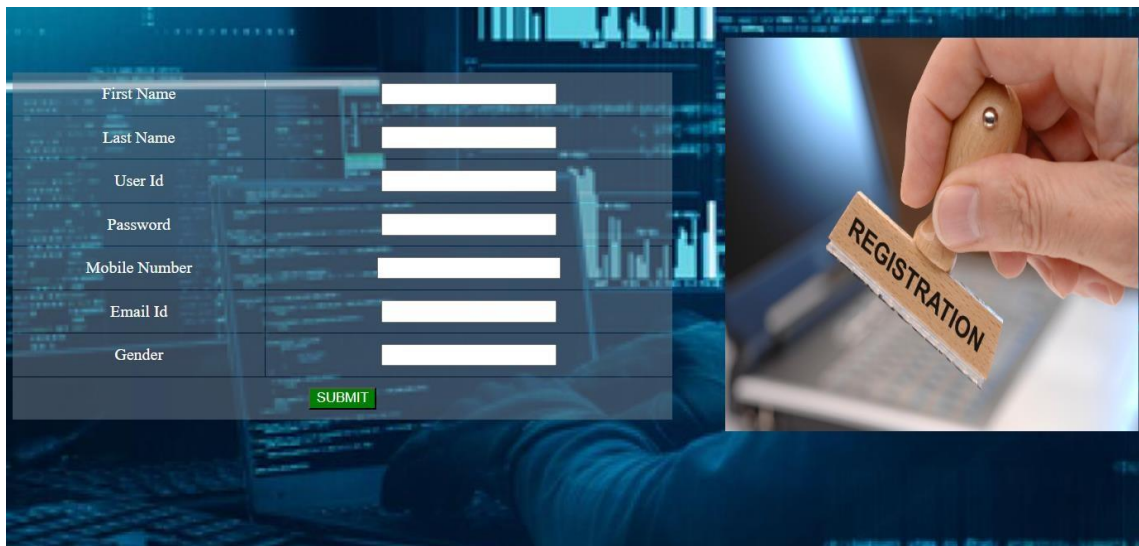


Fig 7.2 Signup page



Fig 7.3 Login page

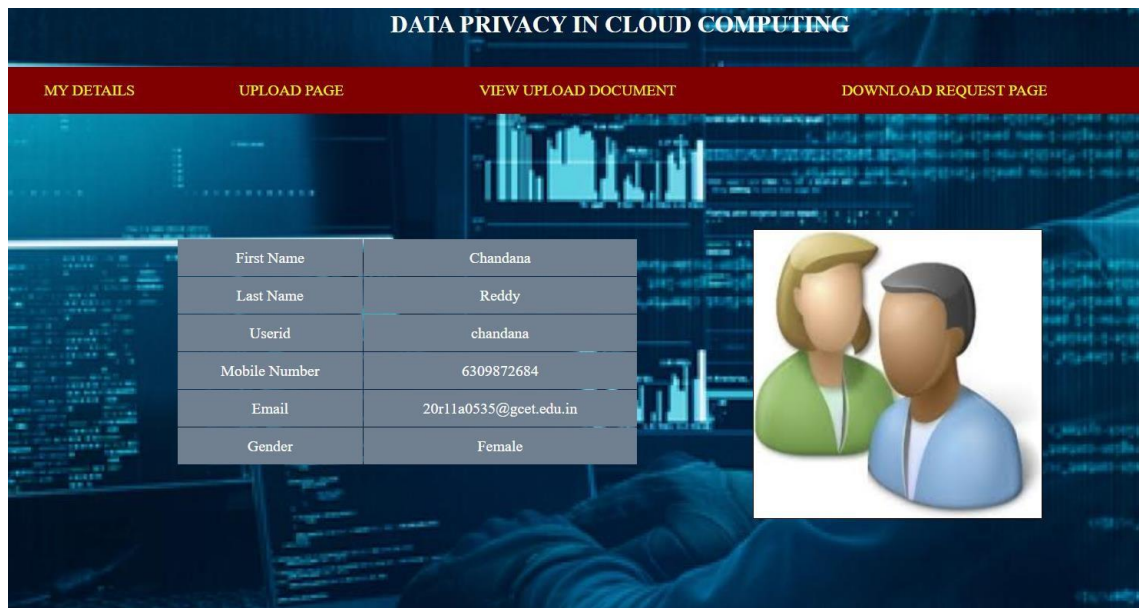


Fig 7.4 Details of Registered User



Fig 7.5 File Upload page

| File Name | Category | Upload File |
|-----------|------------------|---|
| it | IT Asset | Documents_Secure_File_storage_in_Cloud_Computing_using_Hybrid_Cryptography_Algori |
| firstfile | Personal Details | anirudh.jpg |
| mini ppt | Personal Details | mini_review_2_mLKkkIe.pptx |

Fig 7.6 View Uploaded files

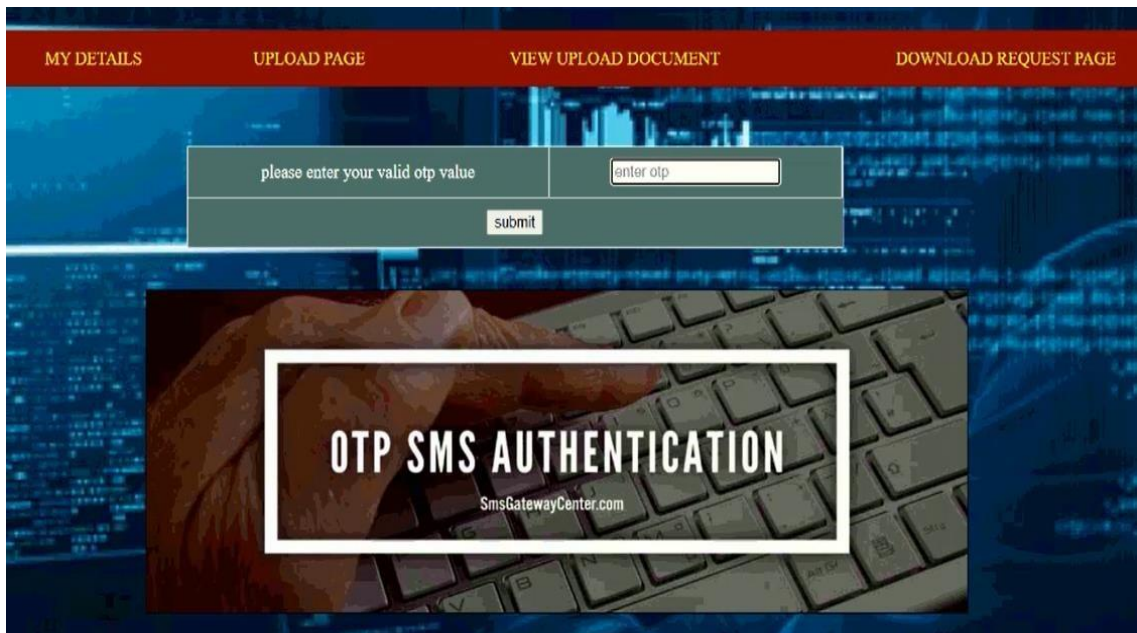


Fig 7.7: Authentication page

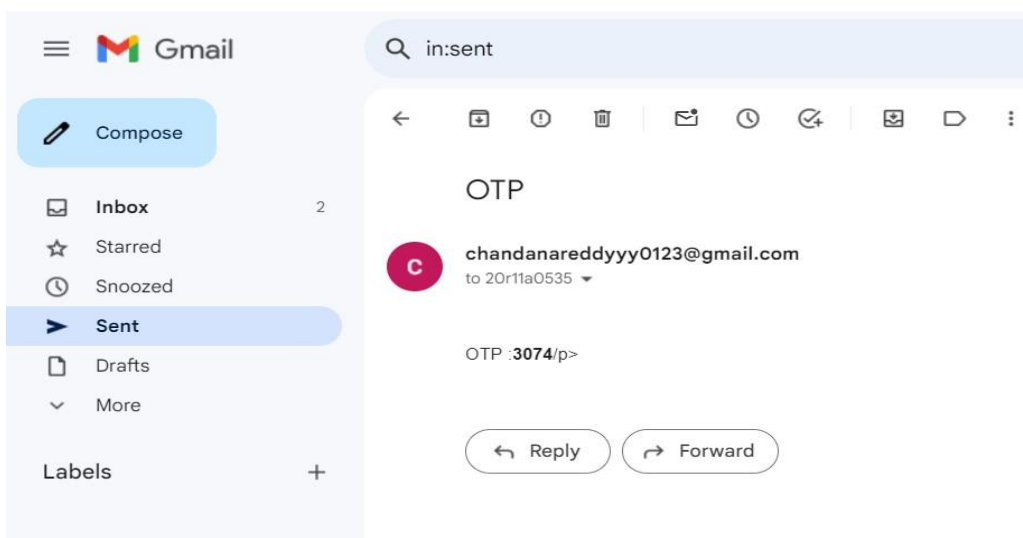


Fig 7.8 OTP in mail

| File Name | Location | Category | Upload File | STATUS |
|-----------|-----------|------------------|-------------------------------------|----------|
| rahul | hyderabad | Personal Details | Farmers_Welfare_management_PPT.pptx | Download |

OK SUCCESSFULLY

Fig 7.9 Authentication page

The screenshots provided depict the sequential flow of the entire project's user interface. Initially, users are presented with the option to either log in or sign up, depending on their requirements. Once authenticated, users can proceed to upload their desired files. A feature allows users to view all the files they have uploaded to our server.

When users wish to download a specific file, they must request an OTP (One-Time Password). This OTP is then generated and sent to the email address associated with their login credentials. Upon entering the correct OTP, users gain access to download and view the file on their system.

In conclusion, the provided screenshots offer a comprehensive overview of the user interface flow for the project, showcasing the seamless journey from authentication to file management and download.

By offering clear options for logging in or signing up, users can easily access the system and begin utilizing its features. The file upload functionality enables users to securely upload their files to the server, while the file management feature allows them to conveniently view all their uploaded files.

The implementation of OTP-based file download adds an extra layer of security, ensuring that only authorized users can access and download files from the server. This enhances data protection and confidentiality, reinforcing the project's commitment to user privacy and security.

The incorporation of OTP-based authentication for file downloads underscores the project's dedication to robust security measures, safeguarding against unauthorized access and data breaches. By leveraging email verification and OTP generation, the system enhances user authentication, ensuring that only legitimate users can access sensitive files.

This additional layer of security instils confidence in users regarding the confidentiality and integrity of their data. With an intuitive user interface and stringent security protocols in place, the project strives to provide a reliable and user-centric solution for secure file storage, addressing the evolving needs and expectations of modern users in an increasingly digital landscape.

Overall, the user interface design promotes user-friendly interactions and efficient navigation, enhancing the overall user experience. With a focus on usability, security, and functionality, the project aims to deliver a seamless and secure file storage solution for users' needs.

Based on the survey it was identified that secure file storage and sharing would not only require confidentiality but also authentication and integrity. To overcome these drawbacks an architecture is proposed which tries to provide a complete solution for securely storing the files. Our project, "Secure File Storage on Local Host using Hybrid Cryptography" within the Django framework, marks a pivotal achievement in data security. By employing hybrid cryptography, it offers a two-tiered defence, safeguarding data during transmission and at rest.

The interface, enriched by Django's features, ensures easy file management, enhancing accessibility. Hosting files locally provides users control and privacy, reducing reliance on third-party cloud services. In conclusion, our project responds to the pressing need for data security, combining advanced encryption with a user-friendly platform. Local hosting and scalability are key strengths. This project serves as a foundation for robust data protection and future developments in secure data management.

Hosting files locally offers users enhanced privacy and control over their data, reducing reliance on third-party cloud services and mitigating associated risks. Furthermore, the project's scalability ensures adaptability to evolving user demands, future-proofing its utility.

Moreover, the architecture incorporates robust authentication mechanisms, bolstering the overall security posture by ensuring that only authorized users can access the stored files. Additionally, the system implements integrity checks to detect and prevent unauthorized modifications to the data. These comprehensive security measures collectively contribute to fortifying the confidentiality, authenticity, and integrity of the stored files, instilling confidence in users regarding the protection of their sensitive information. In summary, our project represents a significant advancement in data security, providing a comprehensive solution that combines advanced encryption techniques with user-friendly functionality. With its emphasis on local hosting, scalability, and robust security measures, it serves as a cornerstone for secure data management practices and lays the groundwork for further innovations in the field.

As technology advances rapidly, it's imperative to explore potential enhancements for our project, "Web-based Application for Secure File Storage on Local Host using Hybrid Cryptography." This project presents exciting opportunities for further development, pushing the boundaries of data security and user experience to new heights.

- **Multi-Platform Compatibility:** As technology advances rapidly, it's imperative to explore potential enhancements for our project, "Web-based Application for Secure File Storage on Local Host using Hybrid Cryptography." This project presents exciting opportunities for further development, pushing the boundaries of data security and user experience to new heights.
- **Integration with Cloud Services:** Integration with cloud services represents another avenue for improvement, enabling users to securely back up encrypted files to cloud storage providers while retaining the benefits of local hosting. This hybrid approach not only enhances data availability but also fortifies data protection with added redundancy.
- **Multiple file Attachment** – enhancing the project's capability to handle multiple file attachments simultaneously would greatly enhance user convenience and efficiency. Empowering users to securely store multiple files in one go streamlines workflow and enhances productivity, aligning with the project's commitment to delivering a seamless and user-friendly experience.

9 REFERENCES

9.1 Book References

[1] Cryptography and Network Security: Principles and Practice" by William Stallings. This book provides a comprehensive introduction to cryptography and network security. It covers various cryptographic techniques, including hybrid cryptography, and how they can be applied to secure data storage.

[2] Network Security and Cryptography: Bernard Menezes cengage learning**Bernard L. Menezes** has been a professor for more than twenty-two years. He has been widely published in national and international workshops, conferences, and journals, including the IEEE Transactions on Computers, the IEEE Transactions on Reliability, and the International Journal of Parallel Computing.

In addition to network security and cryptography, his research interests include parallel computing and smart E-Business.

[3] Cryptography and Network Security: Atul Kahate, McGraw Hill, 3rd Edition 2013. A prolific writer, Kahate is also the author of 38 books on Computer Science, Science, Technology, Medicine, Economics, Cricket, Management, and History.

Books such as Web Technologies, Cryptography and Network Security, Operating Systems, Data Communications and Networks.

[4] Django for Beginners by William S. Vincent: Completely updated for Django 3.1. Django for Beginners is a project-based introduction to Django, the popular Python-based web framework.

Suitable for total beginners who have never built a website before as well as professional programmers looking for a fast-paced guide to modern web development and Django fundamentals.

9.2 Website References

1. Gokulraj, S. and Ananthi, P. and Baby, R. and Janani, E., Secure File Storage Using Hybrid Cryptography (March 11, 2021). Available at SSRN: <https://ssrn.com/abstract=3802668> or <http://dx.doi.org/10.2139/ssrn.3802668>
2. Al-Shabi MA (2019) A survey on symmetric and asymmetric cryptography algorithms in information security. Int J Sci Res Pub 9(3). <http://dx.doi.org/10.29322/IJSRP.9.03.2019.p.8779>
3. Cryptography and Network Security Principles: <https://www.geeksforgeeks.org/cryptography-and-network-security-principles/>
4. Secure-File-Storage-Using-Hybrid-Cryptography <https://github.com/shivang8/Secure-File-Storage-Using-HybridCryptography?search=1>
5. Cryptography is fundamental to the Confidentiality and Integrity of applications and systems. The OWASP [Cheat Sheet](#) series describes the use of cryptography and some of these are listed in the further reading at the end of this section. <https://owasp.org/www-project-developer-guide/draft/04-foundations/04-cryptoprinciples>

9.2 Technical Publishes References

[1] S. Hesham and Klaus Hofmann, "High Throughput Architecture for the Advanced Encryption Standard Algorithm", *IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, pp. 167-170, April 2014.

[2] M. Nagle and D. Nilesh, "The New Cryptography Algorithm with High Throughput", *IEEE ICCCI*, pp. 1-5, January 2014. The author has included the *Improved DES algorithm uses a 112-bit key size for data encoding and decoding.*

Django is a popular open-source web framework written in Python that is primarily used for building web applications. It provides a set of tools and features for web developers to simplify and streamline the development process. While Django is primarily known for web development, it can play a role in this project on secure file storage on a local host using hybrid cryptography, albeit in a more indirect way. In this project, Django can be employed to create a user-friendly web interface for interacting with the file storage system. The framework's authentication and authorization functionalities can be utilized to manage user accounts, enforce access control policies, and authenticate users securely. Additionally, Django's built-in ORM (Object-Relational Mapping) can facilitate interactions with the database for storing user data and file metadata. While Django may not directly handle the cryptographic operations involved in hybrid cryptography, it can provide a solid foundation for building the administrative and user-facing components of the system, enhancing overall usability and manageability. Thus, Django's versatility makes it a valuable asset in complementing the secure file storage solution.

- **Web Interface:** Django can be used to create a web-based user interface for a secure file storage system. We can use Django's built-in features for handling user authentication, managing user accounts, and creating an intuitive web interface for users to upload, download, and manage their files securely. Leveraging Django's built-in features, developers can implement robust user authentication mechanisms, manage user accounts securely, and design an intuitive interface for file upload, download, and management. By utilizing Django's powerful tools and frameworks, developers can streamline the development process and deliver a seamless user experience, ensuring efficient and secure file storage operations for users.
- **RESTful APIs:** Django can also be used to create RESTful APIs to interact with our file storage system. These APIs can handle requests for file uploads, downloads, encryption, decryption, and other functionalities related to hybrid cryptography file storage. Django's flexibility extends to creating RESTful APIs, allowing seamless interaction with the file storage system.

Leveraging Django's capabilities, developers can design APIs to handle various operations, including file uploads, downloads, encryption, decryption, and other functionalities related to hybrid cryptography file storage. These APIs provide a standardized and efficient means for clients to communicate with the system, enabling interoperability and facilitating integration with various applications and platforms. By leveraging Django's framework for API development, developers can ensure scalability, security, and reliability in their file storage system.

- **Security:** Django has a strong focus on security. It provides features like user authentication, data validation, and protection against common web vulnerabilities (such as SQL injection and Cross-Site Scripting). These security features can complement the security measures you implement for our hybrid cryptography system. Django's emphasis on security, including user authentication and protection against common web vulnerabilities, complements the security measures implemented in the hybrid cryptography system. By leveraging Django's robust security features, developers can enhance the overall security posture of the file storage system, ensuring data integrity and protecting against potential threats.
- **Database Management:** Django includes an Object-Relational Mapping (ORM) system, which allows to define and manage data models in Python code. You can use this feature to manage information about files, user accounts, and encryption keys in your secure file storage system. Django's Object-Relational Mapping (ORM) system provides a seamless interface for defining and managing data models using Python code. This feature simplifies database interactions by abstracting away the complexities of SQL queries, allowing developers to work with Python objects instead. By leveraging Django's ORM, developers can define data models for files, user accounts, and encryption keys, facilitating efficient data management within the secure file storage system. The ORM system seamlessly integrates with Django's built-in database migrations, ensuring consistency and ease of data schema evolution as the application evolves over time. Overall, Django's

ORM enhances productivity and maintainability in developing and managing the file storage system.

- Scalability: When a project evolves and needs to handle a larger user base, Django provides tools and practices for scaling web applications. It can help adapt to increased demands on secure file storage systems.

However, it's important to note that Django, as a web framework, is not a direct component of hybrid cryptography. Implementation of hybrid cryptography for secure file storage will likely involve a combination of cryptographic libraries and techniques to ensure data security. Django can serve as a platform for managing the application's overall functionality, user interactions, and data management.

While Django does not directly contribute to hybrid cryptography, it serves as a foundational tool for managing the application's overall functionality, user interactions, and data management.

While cryptographic libraries and techniques handle the encryption and decryption processes, Django provides the infrastructure for implementing and integrating these features into the project. Django's capabilities extend beyond cryptographic operations, offering a robust framework for developing web-based interfaces and managing server-side logic. It provides features for user authentication, session management, data modelling, and API development, which are essential components of a secure file storage system.

By leveraging Django's capabilities alongside cryptographic libraries, developers can create a comprehensive solution that combines strong security measures with user-friendly functionality. This combination allows for the seamless integration of hybrid cryptography features into the project, ensuring both data security and a positive user experience.

In summary, while Django is not a cryptographic library itself, it can be a valuable tool for building the web-based interface and managing the server-side logic for a secure file storage project, which includes hybrid cryptography features.

