Cloud-based Secure File Storage System using Hybrid Cryptography

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

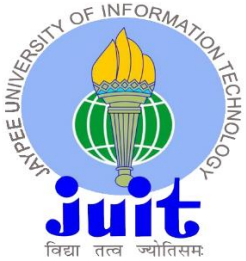
Submitted by

Kanishk Gupta (201188)

Bhavik Chauhan (201120)

Under the guidance & supervision of

Dr. Deepak Gupta



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology,

Waknaghat, Solan - 173234 (India)

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **"Cloud-based Secure File Storage System using Hybrid Cryptography"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of **Computer Science & Engineering and Information Technology**, Jaypee University of Information Technology, Waknaghat is an authentic record of our work carried out by Kanishk Gupta (201188) and Bhavik Chauhan (201120) during the period from August 2023 to May 2024 under the supervision of **Dr. Deepak Gupta**, Assistant Professor (SG), Department of Computer Science and Engineering and Information Technology, Waknaghat.

Kanishk Gupta (201188) **Bhavik Chauhan (201120)**

The above statement made is correct to the best of my knowledge.

Dr. Deepak Gupta
Assistant Professor (SG)
Department of CSE and IT
Jaypee University of Information Technology

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled **'Cloud-based Secure File Storage System using Hybrid Cryptography'** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of our work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Deepak Gupta** (Assistant Professor (SG), Department of Computer Science & Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)
Student Name: Kanishk Gupta
Roll No.: 201188

(Student Signature with Date)
Student Name: Bhavik Chauhan
Roll No.: 201120

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)
Supervisor Name: Dr. Deepak Gupta
Designation: Assistant Professor (SG)
Department: Computer Science & Engineering

Dated:

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have contributed to the completion of this project - "**Cloud-based Secure File Storage System using Hybrid Cryptography**."

First and foremost, we extend our deepest appreciation to our project supervisor **Dr. Deepak Gupta**, whose guidance played a vital role in setting the course of this research. Your valuable insights and support have been essential throughout the entire duration of this project.

We would also like to thank Jaypee University of Information Technology, Solan for providing the necessary resources and environment for completing this project.

A special thanks to our peers who provided constructive input and engaged in discussions, ultimately improving the project's overall quality.

Lastly, we express our gratitude to family and friends for their continuous encouragement and support. Your patience and understanding during the challenges faced during the project, have been invaluable.

This project would not have completed without the combined efforts of all those mentioned above. Thank you for being an important part of this journey.

Kanishk Gupta
Bhavik Chauhan

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

LIST OF FIGURES

ABSTRACT

Given the increasing reliance, on cloud services for storing data it has become crucial to have security measures in place to protect information. This project introduces a cloud based secure file storage system that utilizes a hybrid cryptography approach. By combining asymmetric algorithms this system addresses the limitations of individual methods and enhances overall data security.

The main functionality of this proposed system involves uploading, storing, retrieving and sharing files within a cloud environment. The symmetric key algorithm ensures encryption and decryption of files while the asymmetric key algorithm is used for secure key exchange and user authentication. This hybrid approach not only strengthens data confidentiality, but also offers a scalable and effective solution for cloud-based file storage.

To validate the effectiveness of this system, extensive testing was carried out including performance evaluations and security assessments. The results demonstrate that the system can securely handle types and sizes of files with impact on performance. Additionally, rigorous security analysis was conducted during implementation to ensure resistance against attacks and unauthorized access.

The cloud based secure file storage system presented in this project addresses the growing concerns regarding data security, in cloud environments by utilizing the advantages of cryptography, this system offers a solution, for individuals and businesses looking for a safe and effective way to store and handle their files in the cloud.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

As more and more organizations and individuals move their data to the cloud the issue of security and keeping information confidential has become a concern. This project aims to address these concerns by developing and implementing a cloud based secure file storage system that uses cryptography to enhance the security of stored data.

In the past securing data, in the cloud has relied on either asymmetric cryptography, each with its advantages and limitations. Symmetric key encryption is great for processing of files but faces challenges when it comes to managing keys securely and distributing them. On the other hand, asymmetric key encryption excels in key exchange and user authentication but can be computationally intensive especially with large datasets.

To tackle this challenge effectively this project takes a cryptography approach. By combining the strengths of asymmetric encryption algorithms, the system finds a balance between performance and security. The symmetric algorithm is used for encrypting and decrypting files to optimize efficiency while the asymmetric algorithm handles secure exchange and user authentication to enhance data security.

The goal of this project is to provide a strong yet user solution, for storing files in the cloud. The results of this project add to the increasing amount of information, in the field of security. It provides an efficient solution, for people and companies who want a way to store and handle their files in cloud environments.

1.2 PROBLEM STATEMENT

The rapid advancement of cloud computing has revolutionized the way data is stored and accessed offering convenience and scalability. However, this transition, to cloud-based solutions has also raised concerns regarding the security and privacy of data. Traditional encryption methods may not provide protection against evolving cyber threats.

In efforts to enhance security there has been a predominant reliance on single method encryption techniques. However, these approaches have faced limitations in delivering protection. Symmetric encryption, which is efficient for data transfer lacks the key

management for long term security. On the other hand, asymmetric encryption while strong for exchange may be less suitable for encrypting large scale data. Consequently, previous solutions have left a gap in achieving an approach to securing file storage on cloud platforms.

To address these challenges our project proposes the development of a cloud based secure file storage system that utilizes hybrid cryptography. The goal is to create an adaptable solution that combines the strengths of both symmetric and asymmetric encryption techniques to establish an advanced level of data security. This will ensure that user data stored on cloud platforms remains protected.

The hybrid cryptography approach serves as the foundation, for our project's proposed solution. This method guarantees that information stored in the cloud remains private and cannot be altered, when faced with cyber threats. Additionally, our suggested system will also tackle the issue of protecting data while it is being transmitted and stored. We aim to increase organizations confidence, in using cloud storage for their data by giving importance to both the security of data and the experience of users.

1.3 OBJECTIVES

1) **HYBRID CRYPTOGRAPHY IMPLEMENTATION**

   Develop a system to use two types of encryptions: symmetric and asymmetric in order to make data stored on the cloud more secure. This would make the entire system more reliable.

2) **USER-FRIENDLY INTERFACE**

   A friendly UI with features like upload, access and manage files on the cloud without any hassle is one of the top objectives of our project.

3) **EFFICIENT FILE ENCRYPTION AND DECRYPTION**

   Implement a symmetric key encryption algorithm to facilitate the secure and efficient encryption and decryption of files, ensuring that the system can handle varying file sizes without compromising performance.

4) **TESTING AND VALIDATION**

   Testing the system's security measures, performance and user experience thoroughly to validate its effectiveness and identify potential vulnerabilities. Assess the system's

efficiency and responsiveness to ensure scalability and reliability in real-world usage scenarios. Conduct a thorough security analysis of the implemented system, identifying and addressing potential vulnerabilities and threats. Verify the system's resistance to common cryptographic attacks and unauthorized access attempts.

1.4 MOTIVATION AND SIGNIFICANCE

In an era dominated by the increase of cloud computing, the seamless storage and retrieval of data have become integral to both individual users and organizations. However, the benefits of cloud storage are accompanied by significant concerns regarding the security and privacy of stored information. The motivation behind the development of the "Cloud-based secure file storage system using hybrid cryptography" stems from the imperative need to address these security challenges and provide a robust, efficient, and user-friendly solution for safeguarding sensitive data in cloud environments.

1.4.1 MOTIVATION

- **GROWING DEPENDENCE ON CLOUD SERVICES**
  With the exponential growth in data generation, storage, and sharing, individuals and organizations increasingly rely on cloud services. The convenience offered by these services, however, raises pressing questions about the security of the stored data.

- **HEIGHTENED SECURITY CONCERNS**
  Recent high-profile security breaches and data compromises have underscored the vulnerability of information stored in the cloud. The motivation to enhance cloud security becomes even more critical in the face of evolving cyber threats.

- **LIMITATIONS OF EXISTING SOLUTIONS**
  While various encryption methods are employed to secure data in transit and at rest, existing solutions often face trade-offs between security and performance. Symmetric key encryption may excel in speed but poses challenges in key distribution, while asymmetric key encryption, though secure, can be computationally intensive.

1.4.2 SIGNIFICANCE

- **ENHANCED SECURITY THROUGH HYBRID CRYPTOGRAPHY**

The significance of this project lies in its adoption of hybrid cryptography, marrying the strengths of symmetric and asymmetric encryption. This approach aims to provide a comprehensive security solution that overcomes the limitations of individual cryptographic methods, ensuring robust data protection without compromising performance.

- **OPTIMAL PERFORMANCE AND SCALABILITY**
  The significance of implementing a hybrid cryptography approach is not only in enhancing security but also in optimizing the performance of file encryption and decryption. By carefully balancing the use of symmetric and asymmetric algorithms, the system aims to deliver efficient operations that scale seamlessly with varying workloads.

- **USER-CENTRIC DESIGN**
  The project places importance on user experience by developing an intuitive and user-friendly interface for secure file retrieval, sharing and storage. This ensures that the security measures do not impede the usability of the system, making it widely accessible to a number of users.

- **CONTRIBUTION TO CLOUD SECURITY KNOWLEDGE**
  The findings of this project contribute to the body of knowledge in cloud security by exploring the practical application of hybrid cryptography. Documenting the design decisions, implementation details, and testing outcomes adds valuable insights to the broader discussion on securing data in cloud environments.

In summary, this project is motivated by the pressing need to secure data in cloud storage, and its significance lies in providing a novel solution that combines security, efficiency, and user-friendliness through the integration of hybrid cryptography.

1.5 ORGANIZATION OF PROJECT REPORT

The report is organized as follows:

- Chapter 1 provides us the introduction of the study, along with the problem statement, objectives, significance, motivation and organization of the project.

- Chapter 2 outlines the existing related work in the field of secure storage systems and cryptographic algorithms, it further presents the outputs of their analysis which we compare and use in our project.

- Chapter 3 defines the functional and non-functional requirements of the cloud-based secure file storage system, also based on the detailed analysis of user needs. This chapter also includes the design and architecture of the project, and the implementation of the project.

- Chapter 4 gives us the testing strategy, consisting of the tools, methodologies and any particular testing environments used to perform this project. This chapter also provides detailed test cases, scenarios and outcomes of testing, highlighting any issues encountered with their resolutions.

- Chapter 5 presents the results and evaluation of the project along with comparison to any existing solution, emphasizing improvements and new features.

- Chapter 6 presents the conclusion of the study providing limitations and key findings of this project, along with the discussion of additional features and modifications that can be made in the future aspects.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

The literature survey for the project "Cloud-based secure file storage system using hybrid cryptography" attempts to explore the existing knowledge in the field of cloud computing, secure file storage and hybrid cryptography. The survey includes standard books, scientific journals, reputable websites, and technical research papers from top organizations to ensure inclusion of the latest modifications and advancements in this field, especially in the last five years.

The analysis begins with an in-depth review of the literature related to cloud computing, emphasizing the increased use of cloud services, and the challenges related to data security in cloud environments. It explores established models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) for a brief understanding of the cloud landscape.

At the same time, secure application file storage systems, existing protocols, encryption methods and access control mechanisms are analyzed. Special emphasis is placed on research to address vulnerabilities and threats inherent in cloud-based file storage, and to support proposed hybrid cryptographic solutions.

The second major area of literature research is hybrid cryptography, which evaluates research on combining symmetric and asymmetric encryption algorithms. Notable advances and new approaches in hybrid cryptography are identified and reviewed to understand how they contribute to the balance between performance and security, especially in cloud-based data storage environments.

S. No.	Paper Title [Cite]	Journal / Conference (Year)	Tools / Techniques / Dataset	Results	Limitations
1.	A hybrid elliptic curve cryptography (HECC) technique for fast encryption of data for public cloud security [1]	International Journal of Engineering Research and Technology (2023)	The effectiveness and security of the HECC approach were assessed using a simulated dataset of text and image files.	In the evaluation, the suggested HECC approach produced positive performance and security outcomes.	The proposed HECC technology is not yet ready for commercial use for cloud encryption.
2.	Secure storage - Confidentiality and Authentication [2]	ScienceDirect (2022)	There was no empirical evaluation of the various safe storage systems in the survey.	The survey is a great resource for learning about secure storage ; including threats, techniques, and solutions.	Existing research does not address problems of maintaining secure storage solutions in complex IT settings.
3.	Secure Cloud Storage with Client-Side Encryption using TEE [3]	10th Int. Conference on Cloud Computing and Services Science (2020)	The proposed solution's performance was evaluated using a simulated dataset of 100 MB files.	The proposed solution outperformed standard client-side encryption solutions in terms of performance.	Only Intel SGX-enabled processors are supported by the proposed solution.
4.	A secured cryptographic system based on DNA and a hybrid key generation approach [4]	BioSystems (2020)	A synthetic dataset of text and image files was used to assess the system's performance and security.	In the evaluation, the proposed system achieved good performance and security outcomes.	Suggested system does not cover all characteristics of commercial cryptographic system.
5.	Hybrid Cryptography Algorithms in Cloud Computing: A Review [5]	IEEE 15th International Conference on Electronics, Computer and Computation (2019)	We analyzed several research articles on hybrid cryptography techniques in cloud computing.	Hybrid cryptography algorithms offer improved security and performance over traditional encryption algorithms.	There was no empirical evaluation of the various hybrid cryptography algorithms in the review.
6.	Secure file storage in cloud computing using hybrid cryptography algorithm [6]	International Journal of Engineering Research and Technology (2016)	A simulated dataset of text and image files was utilized to assess the algorithm's performance and security.	In the evaluation, the suggested algorithm achieved good performance and security outcomes.	More real-world testing is needed to assess the performance and security of secure storage technologies.
7.	A Secure Storage Service in the Hybrid Cloud [7]	IEEE 4th International Conference on Utility and Cloud Computing (2011)	A synthetic dataset of files was used to evaluate the performance and security of TrustStore.	In the evaluation. TrustStore achieved good performance and security results.	TrustStore is currently in its early stages and does not yet support all the capabilities of a commercial cloud storage service.

B. R. Rao et al. [1], proposes a new HECC method that is fast and efficient for data encryption over pubic cloud. This approach exploits a hybrid method in which Elliptic Curve Cryptography (ECC), an instance of the public key cryptography, is combined with the fast symmetric key algorithm Blowfish for security and efficiency respectively. Computationally efficient Blowfish is used for bulk-data encryption because it supports key exchange and digital signature creation using ECC. Combination of these allows for higher speed encryptions as well as stronger cryptographic security.

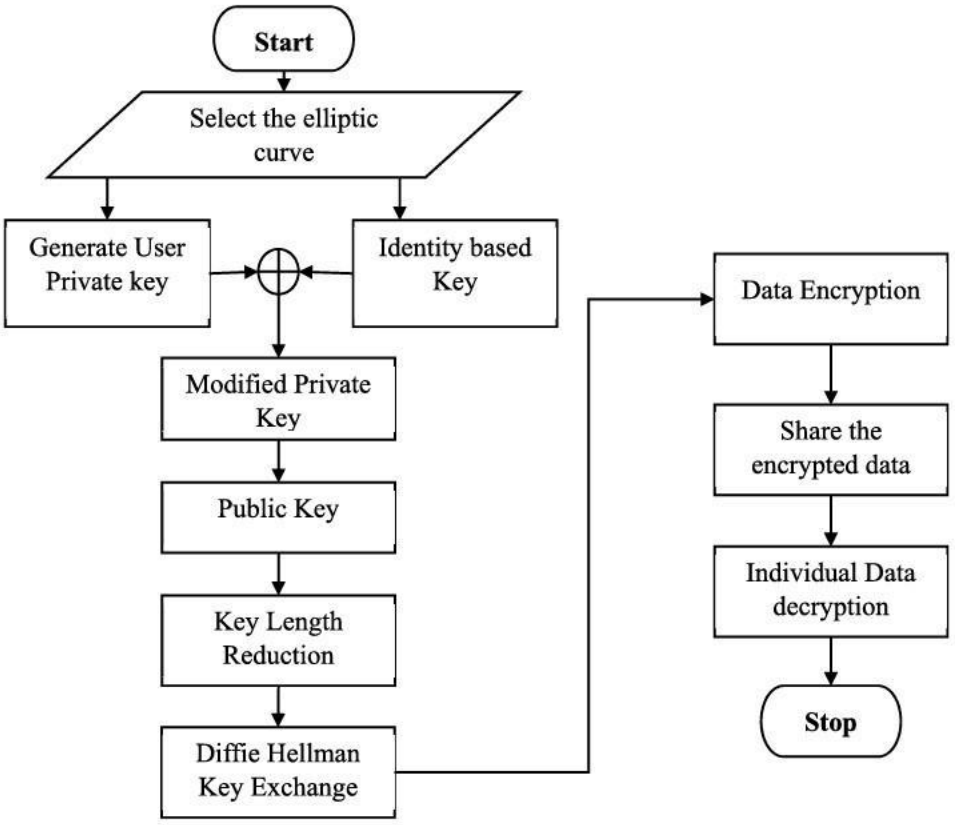


Fig. 2.1: Proposed model flow graph with hybrid ECC, Diffie Hellman and AES

R. Benadjila et al. [2], highlights the limitations of classical approaches for data encryption, especially in consideration of the underlying system-level architectures involved. A crucial aspect is how performance would be affected by having these details ignored in the encryptions scheme used. This section discusses the traditional "Full Disk Encryption (FDE)" approach that is assumed suitable for disk encryption application. This provides for a confidential environment but does not incorporate cryptographic data integrity because certain weaknesses can occur.

M. da Rocha et al. [3], address the growing importance of data confidentiality due to the increasing amount of sensitive data stored in computer systems. Cloud data storage services allow convenient access to data from anywhere, but this convenience raises concerns about data confidentiality when third parties store the data. The author uses Intel SGX which is a technology that provides a trusted execution environment (TEE) within the CPU. The data sealing feature of Intel SGX allows applications to encrypt data before sending it to a storage provider. Even if the storage provider is compromised, the data remains secure because it is encrypted within the enclave. The authors create a virtual file system where applications can store their data. Data sealing using Intel SGX ensures that the data remains confidential. To validate their proposal, they integrate the Cryptomator software (a free client-side encryption tool for cloud files) with an Intel SGX application (enclave) for data sealing. The solution demonstrates feasibility in terms of both performance and security. It can be expanded and refined for practical use and integration with cloud synchronization services.

M. I. Reddy et al. [4], explored a bio-inspired cryptographic system that leverages DNA and a hybrid key generation approach. The scheme draws inspiration from genetic encoding, transcription, and translation processes. Some inverse procedures are used for data encryption and decryption. The focus is on the Central Dogma of Molecular Biology (CMDB). DNA itself serves as the carrier for data instead of traditional digital media (e.g., images, text, or videos). Biological tools related to DNA are used for implementation. The Feistel network structure integrates DNA encoding and DNA operators for cryptographic purposes. The Diffie–Hellman Key Exchange approach is utilized for key generation. Additional modifications are done to enhance the strength of the generated keys. Bidirectional Associative Memory Neural Network (BAMNN) is used for recognizing and restoring sets of keys. The proposed method is compared with traditional cryptographic techniques. It results in a 55% increase in processing time for the encryption process and a 67% increase in processing time for the decryption process.

A. B. Garko et al. [5], explores the use of hybrid cryptographic algorithms in overcoming security challenges associated with cloud computing such as ECC and Blowfish: ECC is among the well-known public-key encryption algorithm due to its efficiency and reliability. The blowfish, which is another of the speedily operated symmetric key encryption algorithms. ECC and Blowfish combine to achieve a perfect balance in the cloud environment with a compromise of optimal security and efficiency.

2.2 KEY GAPS IN THE LITERATURE

In B. R. Rao et al.'s work, it was possible to identify the gaps with this paper but not the gaps' specificity that made these gaps hard to define additional studies needed to close such gaps successfully. If none of these gaps exist, it will be difficult to comprehend the problems specifically tackled in the current work, or in what way the suggested method will enhance the hybrid cryptosystems in cloud security.

In R. Benadjila et al.'s work, FDE schemes are problematic in the sense of maintaining a balance between security and performance. The idea evokes concerns on issues related to data integrity within length-preserving encryption schemes, as well as the complications ensuing during authenticated encryption. The data remain at risk with many vulnerabilities because existing FDE solutions do not provide cryptographic data integrity protection. Intelligent attacks target the primitive authorization techniques.

In M. da Rocha et al.'s work, server-side encryption is traditionally considered as a possible source of key compromise. Lack of a centralized approach can make it difficult to identify suitable hybrid cryptography techniques for cloud-based application.

In M. I. Reddy et al.'s work, it fails to explain in detail any gaps that have been found in the existing cryptographic systems and how such refinements or enhancements would be achieved.

In A. B. Garko et al.'s work, the paper pinpoints the wearisome side of key management and possibly the lapse in effectiveness of key managements systems. Hybrid cryptography may have an effect on cost of computations and may also degrade performances in cloud environments. Selecting suitable hybrid cryptography methods for cloud-based application can be difficult due to standardization challenges.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

3.1.1 Software Requirements

- Programming Language: Python 3.0
- Cloud Service Provider: Render
- Integrated Development Environment (IDE): Visual Studio Code (VS Code)
- Git / Github

3.1.2 Hardware Requirements

- Random Access Memory (RAM): 4 GB or above
- Central Processing Unit (CPU): 2.4 GHz Processor and above
- Operating System (OS): Windows / Linux
- User devices: Windows 7 or above

3.1.3 Other Requirements

- For testing and deployment: Virtual machines or cloud instances
- Internet connectivity

3.1.4 Functional Requirements

- **User Authentication:** Users need a secure way to sign up and log in to the system. This ensures that only authorized individuals can access their files. To enhance security, a two-factor authentication mechanism will be implemented. This means users will need to provide two forms of verification, such as a password and a unique code sent to their mobile device, before gaining access. It adds an extra layer of protection against unauthorized access.

- **File upload and download:** Users should be able to upload files securely from their devices to the cloud-based storage system. This ensures that sensitive data remains protected during transit. Similarly, when files are downloaded, proper access permissions should be enforced. This means that only users with the necessary authorization can access and download specific files, maintaining data confidentiality and integrity.

- **Encryption mechanism:** To ensure end-to-end security, all files stored in the system will be encrypted. This means that even if someone gains unauthorized access to the storage infrastructure, they won't be able to view or tamper with the contents of the files. Hybrid cryptography will be adopted to enhance key management security. This involves using a combination of symmetric and asymmetric encryption algorithms to encrypt and decrypt data. It adds an extra layer of protection by making it more difficult for attackers to compromise the encryption keys.

- **Access controls:** Different users will have different roles and permissions within the system. For example, an administrator might have full access to all files and settings, while a regular user might only have access to their own files. Access controls will be implemented to enforce these permissions, ensuring that users can only perform actions that are allowed based on their role. This helps prevent unauthorized access and misuse of sensitive data.

- **File management:** Users should be able to perform basic file management tasks such as renaming and deleting files. This gives them greater control over their data and helps them organize their files more effectively. However, these actions should only be allowed for users who have the necessary access permissions. For example, a user should only be able to delete their own files or rename files they own. This helps prevent accidental or malicious deletion of important data.

3.1.5 Non - Functional Requirements

- **Security:** The system must ensure high levels of security to protect user data from unauthorized access, modification, or theft. This includes measures such as encryption of data both in transit and at rest, robust authentication mechanisms, and access control policies.

- **Performance:** The system should provide efficient performance, ensuring that file uploads, downloads, and access operations are fast and responsive, even under heavy loads. This includes considerations for latency, throughput, and scalability to handle increasing user demands.

- **Reliability:** Users rely on the system to store their important files securely. Therefore, the system must be highly reliable, with minimal downtime and data loss. This involves

measures such as regular backups, redundancy in storage and network infrastructure, and robust error handling mechanisms.

- **Scalability:** As the number of users and amount of data stored in the system grows, it should be able to scale seamlessly to accommodate this growth. This includes horizontal scalability, allowing for the addition of more servers or resources to handle increased demand, as well as vertical scalability to handle individual file sizes and complexities.

- **Interoperability:** The system should be compatible with a wide range of client devices and operating systems, allowing users to access their files from various platforms seamlessly. This involves adherence to industry standards for file formats, communication protocols, and APIs.

- **Usability:** The system should be easy to use, with a user-friendly interface that allows users to easily upload, download, and manage their files. This involves considerations for intuitive design, clear navigation, and helpful error messages.

- **Auditability:** There should be mechanisms in place to track and audit user activities within the system, providing visibility into who accessed what files and when. This helps in detecting and investigating any security incidents or compliance violations.

- **Maintenance and Support:** The system should be easy to maintain and support, with regular updates and patches to address security vulnerabilities and improve performance. Additionally, there should be timely and responsive technical support available to assist users with any issues or concerns.

3.2 STAKEHOLDERS

3.2.1 End User

- File Management: It enables end users, among other things, to safely upload, download, organize and share it.
- Security Concerns: The users expect heavy duty encryption mechanisms with high level of protection on access control.
- Usability Expectations: Such end users look for a simple environment that allows them to navigate quickly through files.

3.2.2 Developers

- System Architecture: Developers create the system architectural, making scalability, efficiency and security of a huge concern. This entails creating a physical arrangement of the system comprising all the necessary components as they are related to each other.
- Coding and Implementation: This team develops the system by coding, integrating key features, and adding security in every stage it runs. The second phase is where the architected framework is transformed into a working, tangible system.
- Maintenance and Updates: Developers assume the role of continuous system maintenance after implementation. It involves fixing any uncovered errors and performing frequent upgrading aimed at improving system operation and protection.

3.2.3 System Administrators

- User Management: User accounts, access permissions, and authentication are managed by administrators.
- Security Oversight: They supervise security of the system from any possible risks.
- System Monitoring: The role of administrators includes tracking system efficiency and performance, resolving instances of downtimes as well as upholding the sanctity of the database.

3.2.4 Regulatory Bodies

- Compliance Assurance: The system must be in accordance with the local and international laws on data protection under regulatory bodies.
- Privacy Oversight: They check and ensure that user data is safe by evaluating and confirming system's privacy measures.
- Legal Compliance: Legal considerations involved in data storage, personal policies, and users' laws are handled by regulatory agencies.

3.3 FLASK FRAMEWORK

Python Flask, a lightweight and flexible web framework, has been used in this project to provide the backend functionality and web interface. Flask is ideal for developing web apps because of its simplicity, ease of use and huge ecosystem of extensions. Here is a full description of how Python Flask was utilized in the project:

- Backend Development: Flask provides a strong basis for building the web application's backend logic. Flask allows developers to build routes, handle HTTP requests, and run business logic to process data and communicate with databases.
- Routing: Flask allows developers to create routes that convert URL paths into Python functions known as view functions. These view routines process incoming requests and dynamically produce answers based on the specified URL. Routes are specified with decorators, making it simple to organize and manage the application's URL structure.
- HTTP request handling: Flask has built-in support for handling a variety of HTTP requests, such as GET, POST, PUT, and DELETE. Developers may create route handlers that perform various actions depending on the kind of request received from the client.
- Deployment: Flask applications may be readily deployed across a variety of hosting settings, including standard web servers, cloud platforms, and containerized systems. Flask's lightweight design and few dependencies make it ideal for deployment on platforms like as Render, Heroku, AWS, Google Cloud Platform and Docker containers.

Overall, Python Flask is a robust and adaptable framework for constructing web applications, giving developers the freedom and control they need to create bespoke solutions based on individual project needs. Flask played an important role in this project by providing backend logic, managing HTTP requests, generating dynamic content, and integrating with other technologies to build a safe and user-friendly file storage system.

3.4 HOSTING SERVICE

- Accessibility: Using the web interface of Render, the application can be reached through a web browser from any device that is connected to the internet including desktop computers, laptops, tablets, and smartphones. Hence, the project becomes more accessible, thus allowing the users to interact with the secure file storage system without any hassle.
- Scalability: Render, a cloud hosting service, has a scalable infrastructure that is the major feature which enables the application to adapt to the changes in the traffic and user demand. When the user base grows or at the time of peak activity, the app can quickly boost up to the demand and therefore, it can work its best and give a fast response.
- Reliability: The service of Render offers a reliable hosting environment with the extra of redundancy, the features of the failover, and the automatic backups. Therefore, the

possibility of the high availability and uptime for the web interface is enhanced which in turn reduces the downtime and interruptions in user access.

- Security: The hosts of the cloud services are the ones who use the strong security systems to defend the applications and data which are being hosted. The features of this phenomenon include, for instance, encryption, firewalls, intrusion detection, and regular security updates. The web interface on Render is the platform of the project that is the protector of the users' data and meets the privacy and regulatory requirements.
- Ease of Deployment: The implementation of the web interface on Render narrows the deployment process, making the server setup, configuration and maintenance tasks obsolete. The easy and quick launching of updates of the application caused by the intuitive deployment workflows and the integration with version control systems like Git is what the render provides.

In general, the use of the web interface for the project on the cloud hosting service Render makes the file storage system more accessible, scalable, reliable, secure, and easy to use, which means that the secure file storage system is available and easy to use for the users all over the world. This cloud deployment model is a major stride in the accomplishment of the project's aims of providing a secure and convenient solution for the management and protection of the sensitive data in the cloud.

3.5 CRYPTOGRAPHY

Converting information into text so that only the intended user can access it, is known as cryptography. This process involves using techniques such, as scrambling words employing code words or utilizing highly efficient mathematical methods. There are two categories of algorithms used for encryption:

1) **Symmetric Algorithms:** In this algorithm, the same key is used to both encrypt and decrypt the message. For example, if a user wants to send a message to someone and ensure that nobody else can read it, they can encrypt the data using a secret key which is then shared with the intended receiver. The receiver can then use this key to decipher the encrypted data. However, it is important to note that all users who need access to the data must have this shared key.
2) **Asymmetric Algorithms:** These algorithms utilize two keys for encrypting and decrypting the data intended to share. A user encrypts the data using one key called the public key, which is accessible to anyone over the internet. The receiver decrypts the

message using a key known as the private key, that belongs to the receiver only and can be used exclusively by them to decrypt the message.

3.6 HYBRID CRYPTOGRAPHY

When we talk about hybrid cryptography, we are referring to the usage of two or more cryptography techniques. This approach strengthens cloud security as well as ensures the protection of data and privacy.

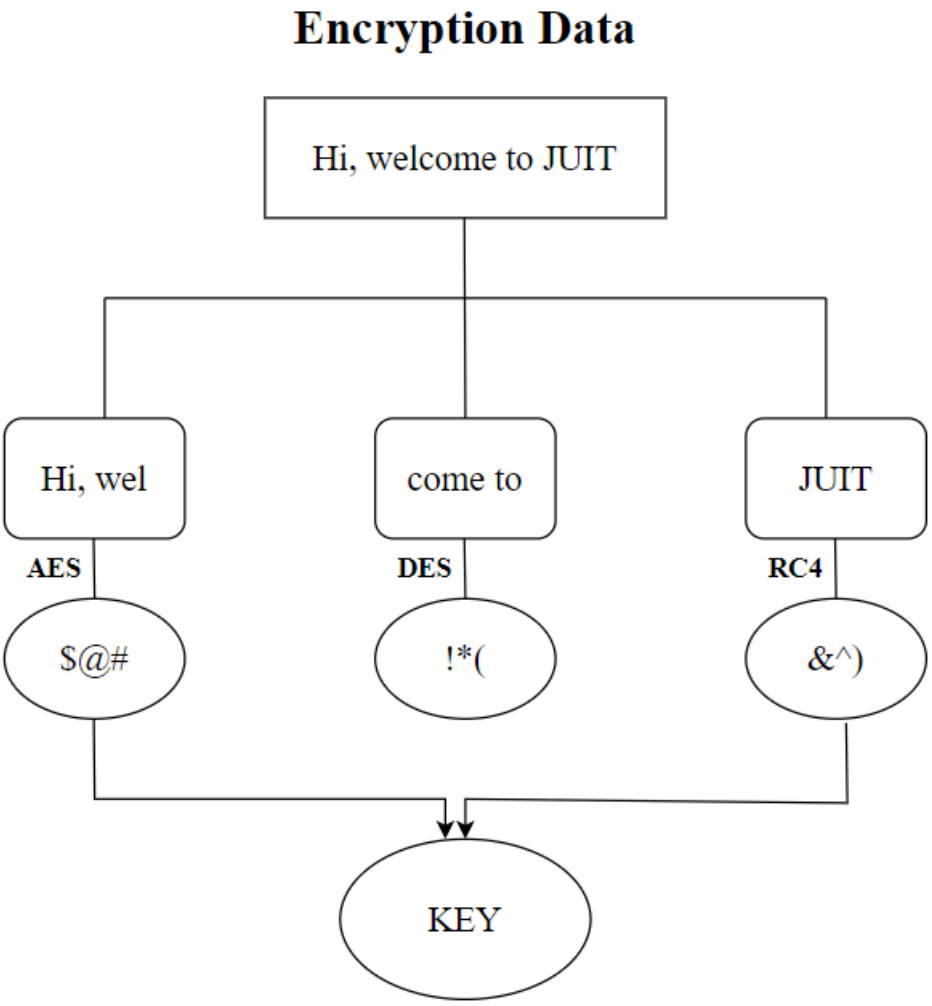


Fig. 3.1: Hybrid Cryptography

3.6.1 NEED OF HYBRID CRYPTOGRAPHY

In the ever-evolving landscape of cybersecurity, the quest for robust and efficient encryption methods remains a constant endeavor. While traditional encryption techniques have served

as stalwart guardians of data security for decades, the emergence of hybrid cryptography represents a significant leap forward in fortifying our digital defenses.

Traditional encryption methods, such as symmetric and asymmetric cryptography, have long been the cornerstone of data protection strategies. Symmetric encryption relies on a single shared key for both encryption and decryption, offering simplicity and efficiency in securing data. On the other hand, asymmetric encryption employs a pair of keys, public and private, for encryption and decryption, respectively, providing enhanced security but with increased complexity.

While traditional encryption methods have proven effective in safeguarding sensitive information, they are not without their limitations. Symmetric encryption, while efficient, poses challenges in key management and distribution, especially in large-scale deployments. Asymmetric encryption, while offering stronger security, can be computationally intensive and impractical for certain applications.

Hybrid cryptography represents a harmonious marriage of the strengths of symmetric and asymmetric encryption techniques, while mitigating their respective weaknesses. By combining the efficiency of symmetric encryption with the robust security of asymmetric encryption, hybrid cryptography offers a versatile and scalable solution for securing data in diverse environments.

The primary advantage of hybrid cryptography lies in its ability to strike a balance between security and efficiency. By leveraging symmetric encryption for data encryption and asymmetric encryption for key management, hybrid cryptosystems achieve a delicate equilibrium that enhances both security and performance. This approach not only ensures the confidentiality and integrity of data but also streamlines key management processes, thereby reducing overhead and complexity.

One of the key benefits of hybrid cryptography is its ability to facilitate secure communication between parties. By encrypting data with a symmetric key and securely transmitting the key using asymmetric encryption, hybrid cryptosystems enable seamless and secure exchange of information over untrusted networks. This ensures that sensitive data remains protected from eavesdroppers and unauthorized access, even in hostile environments.

In an era characterized by escalating cyber threats and sophisticated attacks, the need for robust encryption solutions has never been greater. Hybrid cryptography emerges as a compelling choice for addressing the evolving security challenges of the digital age. Its ability to combine the best of both worlds – the efficiency of symmetric encryption and the security of asymmetric encryption – makes it a versatile and indispensable tool in the modern cybersecurity arsenal.

In conclusion, the adoption of hybrid cryptography represents a paradigm shift in data security, offering a potent blend of security, efficiency, and versatility. By harnessing the strengths of both symmetric and asymmetric encryption techniques, hybrid cryptosystems empower organizations to safeguard their most valuable assets and uphold the confidentiality and integrity of their data in an increasingly interconnected world. As we continue to navigate the complex landscape of cybersecurity, hybrid cryptography stands poised to play a pivotal role in shaping the future of digital security.

3.7 PROJECT DESIGN

FLOW DIAGRAMS

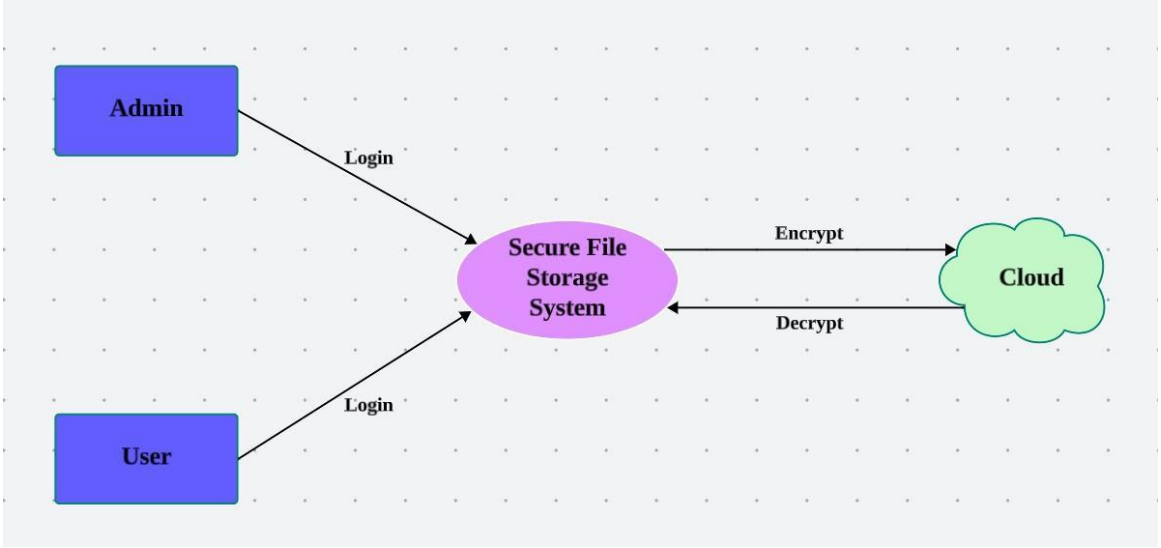


Fig. 3.2: Control Flow Diagram (CFD) for System

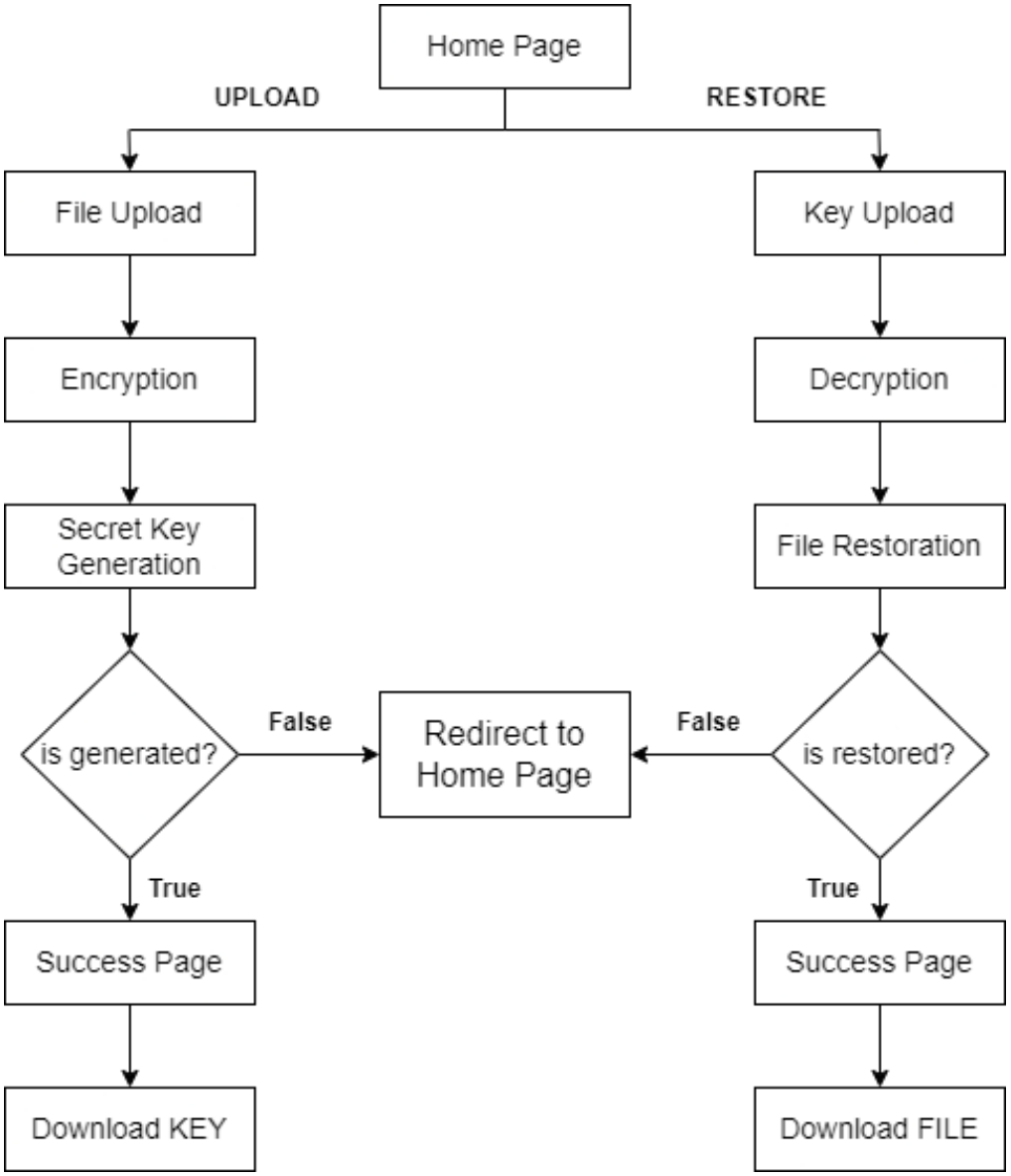


Fig. 3.3: Data Flow Diagram (DFD) for System

The project starts with the home page being displayed to the user to upload a file or to download the original file after restoration.

Uploading: If a user uploads a file, and clicks the Submit button, the success page is displayed from where the user can download the encrypted secret key of the file that they uploaded.

Restoring: If the user possesses an encrypted secret key and uploads it using the Restore button on the home page, upon submission, the user gains access to download the original file which had been uploaded initially.

In this process, once the file is uploaded for encryption and the secret key is downloaded or vice versa, the file storage system will have no record of either the file or the key, ensuring absolute secrecy and data security.

3.8 PROJECT ARCHITECTURE

3.8.1. User Authentication

Objective: Ensure secure user authentication.

Implementation

- Industry Standard Protocols: Use well known authentication protocols like OAuth and OpenID to ensure that the authentication process is secure and uniform.
- Multi-Factor Authentication (MFA): Strengthen security through multi-factor authentication whereby users have to give more than one form for verification purposes with the objective of thwarting unauthorized access.

3.8.2. File Management

Objective: Provide a safe platform for users to upload, download, arrange, and send documents online.

Implementation

- User-Friendly Interface: A design of an intuitive interface should be made incorporating functionalities such as drag and drop in order to enable for smooth management of user's files.
- Real-time Synchronization: Provide a real time synchronization that gives accurate information, and ensures that whatever changes a user does will be updated in the entire system.
- Version Control: Implement version control tools to monitor and manage various file versions; a safety measure that mitigates the risk of losing or damaging data.

3.8.3. Encryption Service

Objective: Ensure that stored file data remain hidden and not accessible to unauthorized persons by using strong encryption.

Implementation

- Hybrid Cryptography: Employ mixed-encryption methods comprising both symmetric and asymmetric approaches for effective and safe data encryption.

- Strong Encryption Algorithms: Make use of strong encryption algorithms such as AES that can provide a tough barrier to the access of stored files that have confidential information, preventing other entities that do not have permission from reading the sensitive data.

3.8.4. Access Control

Objective: Manage user access permissions to files and folders.
Implementation

- Role-Based Access Control (RBAC): Introduce RBAC and allow access rights according to defined roles with permissions required for duties only.
- Access Tokens: The process can be enhanced by using access tokens to identify as well as authorize users increasing the security of access control.
- Fine-grained Access Controls: Putting in place the finer grained access control mechanism that enables the administrative staff to assign specific permissions for file and folder at an increased granular level is very crucial to them.

3.9 IMPLEMENTATION

A comprehensive systemic approach has been applied in the implementation of the cloud-based secure file storage system, which guarantees the reliability, safety as well as user friendliness of the platform. A specific programming language, together with a suitable web framework, will be applied for backend development that involves server logic and database operations. A strong DBMS has been chosen as the backend database to reliably store and access the user information.

For better safety, we will use end-to-end encryption methods such that files remain safe while stored or in transit. Such hybrid encryption techniques will strike a good balance between data security and computer efficiency. In order to prevent unauthorized access of any sort, RBAC will be used to create permissions for users that will enable administrators to specify exact permission levels for every single file and folder on the network.

Users' experience and how it would be responsive and incorporates the use of a frontend framework in the frontend development. User logins will be also protected using secure authentication methods. The system will involve using cloud storage API in dealing with a preferred cloud service provider's storage system making it very scalable and reliable.

Implementing continuous integration and deployment pipelines together with containerization tools, will enable automatic testing and consistent deployments. These include monitoring tools with system performance, user activities and security risks in mind. The documenting and observance of security best practices throughout the process shall enable us develop a robust, secure and easily maintained cloud-based secure file storage system.

3.9.1. TOOLS USED

- Programming Languages:
  Backend Development: The server logic should be handled using a server-side language like Flask and Python.
  Frontend Development: The user interface should be built using languages such as HTML, CSS, JavaScript etc.
- Database Management System (DBMS): Choose a credible DBMS such as MySQL, postgreSQL or MongoDB depending with your data model and demands.
- Cloud Service Provider: Select your cloud vendor from among Render, AWS, Azure or Google Cloud for scalable and reliable cloud storage.
- Encryption Libraries: Use encryption libraries such as Fernet to secure the data prior its storage.
- Authentication and Authorization: Authenticate user logins using authentication libraries or frameworks such as OAuth and JWT to control access.
- Web Development Tools: Use some development codes such as Visual Studio Code, Sublime Text or Atom.
- Version Control: Use version control systems for collaborative development (hosting on GitHub, GitLab or Bitbucket).

3.9.2. TECHNIQUES USED

- Encryption Techniques: Enforce the use of end-to-end encryption while storing and transmitting files. Use hybrid encryption for some compromise between strength of security and of calculation.
- Access Control: Use Role-Based Access Control to administer access privileges of users. Enforce strict granular access control on file level only.
- File Storage: Interact with your preferred cloud provider’s storage system using cloud storage APIs. Use multi-layer approaches in storing data.

- User Interface Design: Ensure that you use responsive design principles whenever designing a user accessible and interactive screen. Dynamic and interactive user interfaces may be achieved using frontend frameworks such as React, Angular, or Vue.
- Testing Techniques: Use frameworks such as Pytest (python) in performing unit testing on individual components. Ensure coherence in the actions of system components by performing integration testing. Conduct user acceptance testing with actual users and check that the system works correctly under realistic conditions.
- Deployment Strategies: Implement CI and CD pipelines with automated testing and deployment. Adopt containerization tools such as Docker consistent deployment in differing settings.
- Monitoring Tools: Adopt monitoring devices such as Prometheus, Grafana towards assessing system performance, user activities, as well as security.
- Security Best Practices: Adhere to security best practice during the development phase involving a number of security issues such as secure coding standards and regular security audits.
- Collaboration Tools: Communicate efficiently with the development team through such tools as Slack, Microsoft Teams, and Jira.

3.9.3. CODE IMPLEMENTATION

```
8   def Algo1(data, key):
9       f = Fernet(key)
10      target_file = open("raw_data/store_in_me.enc","wb")
11      secret_data = f.encrypt(data)
12      target_file.write(secret_data)
13      target_file.close()
14
```

Fig. 3.4: Implementation of Fernet Encryption Algorithm

In this code snippet, the fernet encryption algorithm has been implemented. It takes the data which needs to be encrypted along with the encryption key as parameters. With the help of fernet object's 'encrypt()' method, the function generates the ciphertext, which is then written to the target file.

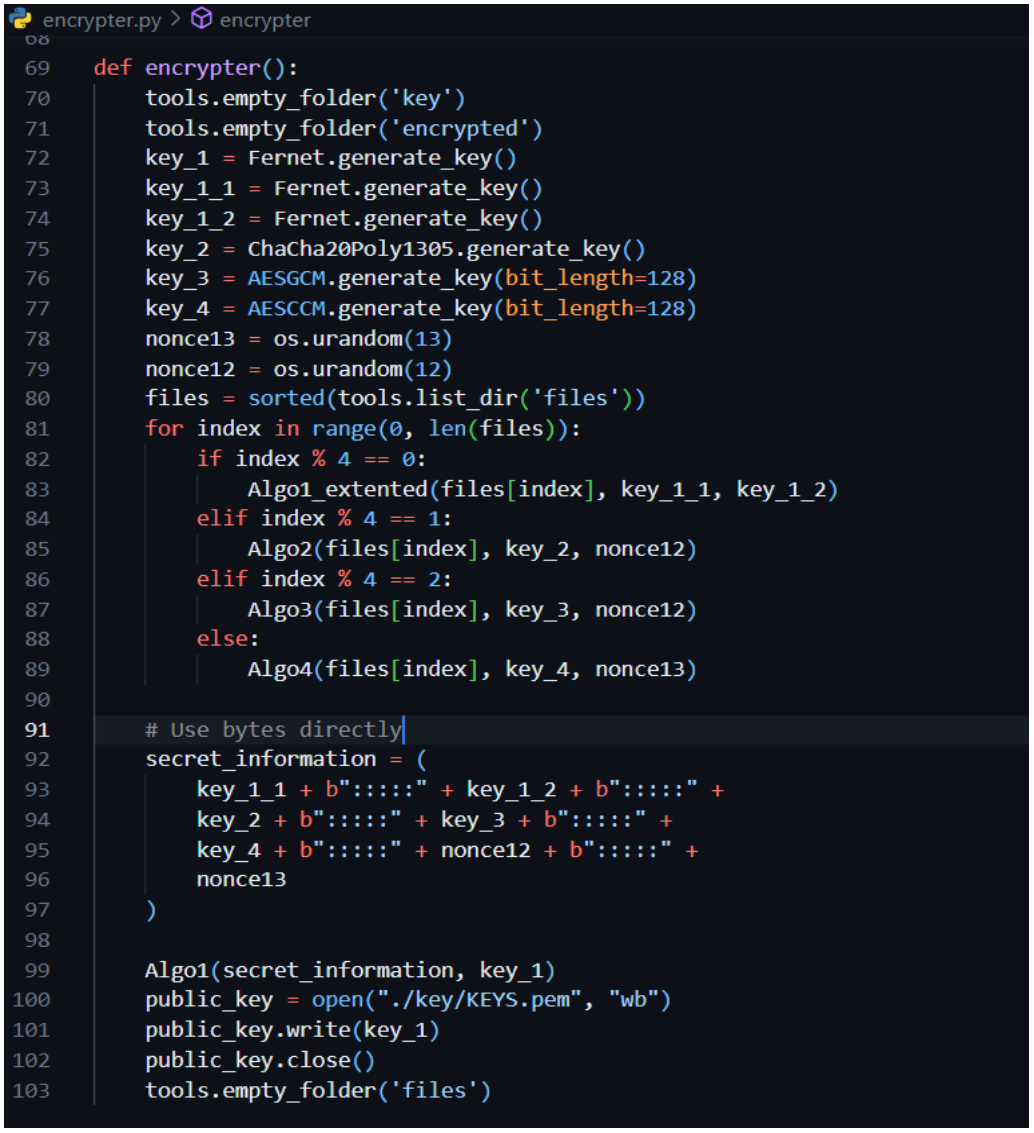


Fig. 3.5: Implementation of encrypter function

In this code snippet, the function for encrypting the files uploaded by the user has been implemented. The function iterates over the files in the designated folder, applying specific encryption algorithm on each file. Following encryption, the generated keys are concatenated into a composite byte string. This composite byte string undergoes a final encryption process using a master key, resulting a public key stored in the file ‘KEYS.pem’ which is essential for decryption.

```
def Algo1(key):
    f = Fernet(key)
    target_file = open("raw_data/store_in_me.enc", "rb")
    secret_data = b""  # Initialize as byte string
    for line in target_file:
        secret_data += line  # Concatenate byte strings directly
    target_file.close()
    data = f.decrypt(secret_data)
    return data

```

Fig. 3.6: Implementation of decryption algorithm

In this snippet, the decryption algorithm to decrypting the file has been implemented. This function accepts the decryption key as a single parameter. Using the decryption key supplied, the function creates a Fernet object. The method starts with an empty byte string called secret_data and iterates over each line of the file, concatenating the byte strings to reconstruct the encrypted data. Using the Fernet object's decrypt() method, the function decrypts the concatenated byte string secret_data, returning the original plaintext data.

```python
def decrypter():
    tools.empty_folder('files')
    key_1 = b""  # Initialize as byte string
    list_directory = tools.list_dir('key')
    filename = './key/' + list_directory[0]
    with open(filename, "rb") as public_key:
        for line in public_key:
            key_1 += line  # Concatenate byte strings directly
    secret_information = Algo1(key_1)
    try:
        # Attempt decoding with utf-8
        list_information = secret_information.decode('utf-8').split(':::::')
    except UnicodeDecodeError:
        # Fallback to latin-1 encoding if utf-8 fails
        list_information = secret_information.decode('latin-1').split(':::::')
    key_1_1 = list_information[0]
    key_1_2 = list_information[1]
    key_2 = list_information[2]
    key_3 = list_information[3]
    key_4 = list_information[4]
    nonce12 = list_information[5]
    nonce13 = list_information[6]
    files = sorted(tools.list_dir('encrypted'))
    for index in range(0, len(files)):
        if index % 4 == 0:
            Algo1_extented(files[index], key_1_1, key_1_2)
        elif index % 4 == 1:
            Algo2(files[index], key_2, nonce12)
        elif index % 4 == 2:
            Algo3(files[index], key_3, nonce12)
        else:
            Algo4(files[index], key_4, nonce13)
```

Fig. 3.7: Implementation of decrypter function

In this code snippet, the function for decrypting the files using the keys uploaded by the user has been implemented. The encrypted data is acquired by calling the Algo1() function and providing the encryption key as a parameter. The encrypted data, which is saved as a byte string, is then decoded to reveal individual encryption keys. To handle decoding problems, the method uses a try-except block, first attempting to decode the data with UTF-8 encoding and then falling back to Latin-1 encoding if that fails. This decryption procedure restores the original content of encrypted files, allowing users to view and use their data safely.

To provide a user-friendly interface for users, we integrated the algorithms into a web application using Flask, a micro web framework for Python.

```
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['UPLOAD_KEY'] = UPLOAD_KEY
```

Fig. 3.8: Initializing the Web Interface

In this code snippet, the flask app is initialized which is essential for the web interface of the project.

Using Flask, we have created various Routes which are listed below

- Default route for the flask application

```
@app.route('/')
def index():
    return render_template('index.html')

```

Fig. 3.9: Route for Default flask application

Here, in this snippet the default route for the web application has been implemented. It renders the homepage of the webpage ‘index.html’

```
<div class="container">
    <div class="header">
        <h1>Secure File Storage using Hybrid Cryptography</h1>
    </div>
    <div class="buttons">
        <a href="upload"><button>Upload</button></a>
        <a href="download"><button>Restore</button></a>
    </div>
    <div class="project">
        <h4>A Project by:</h4>
        <h5>Kanishk Gupta<br>B.Tech (CSE)<br>201188</h5>
        <h5>Bhavik Chauhan<br>B.Tech (CSE)<br>201120</h5>
    </div>
</div>
```

Fig. 3.10: HTML code for index.html

- Route for uploading files

```
# after clicking on upload
@app.route('/upload')
def call_page_upload():
    return render_template('upload.html')
```

Fig. 3.11: Route for uploading files

Here, in this snippet the route for upload file is implemented, and it also renders the upload.html.

```
57
58  <div class="container">
59      <div class="header">
60          <h1>Secure File Storage using Hybrid Cryptography</h1>
61      </div>
62      <div class="form-container">
63          <form action="/data" method="POST" enctype="multipart/form-data">
64              <input type="file" name="file" required/>
65              <br><br><input type="submit"/>
66          </form>
67      </div>
68      <div class="project">
69          <h4>A Project by:</h4>
70          <h5>Kanishk Gupta<br>B.Tech (CSE)<br>201188</h5>
71          <h5>Bhavik Chauhan<br>B.Tech (CSE)<br>201120</h5>
72      </div>
73  </div>
74
```

Fig. 3.12: HTML code for upload.html

- Route for downloading files

```
55  @app.route('/download/')
56  def downloads():
57      return render_template('download.html')
```

Fig. 3.13: Route for downloading files

Here, in this snippet the route for download file is implemented, and it also renders the download.html.

```
56  <body>
57
58  <div class="container">
59      <div class="header">
60          <h1>Secure File Storage using Hybrid Cryptography</h1>
61      </div>
62      <div class="form-container">
63          <form action="/download_data" method="POST" enctype="multipart/form-data">
64              <input type="file" name="file" required/>
65              <br><br><input type="submit" value="Download"/>
66          </form>
67      </div>
68      <div class="project">
69          <h4>A Project by:</h4>
70          <h5>Kanishk Gupta<br>B.Tech (CSE)<br>201188</h5>
71          <h5>Bhavik Chauhan<br>B.Tech (CSE)<br>201120</h5>
72      </div>
73  </div>
74
75  </body>
```

Fig. 3.14: HTML code for download.html

- Route for encrypting the file

```
@app.route('/return-key/My_Key.pem')
def return_key():
    list_directory = tools.list_dir('key')
    filename = './key/' + list_directory[0]
    # return send_file(filename, attachment_filename='My_Key.pem')
    return send_file(filename)

```

Fig. 3.15: Route for encrypting the file

This route is responsible for returning the encryption key file (My_Key.pem) to the user. It locates the key file in the key directory and delivers it as a downloaded attachment.

- Route to download decrypted file

```
@app.route('/return-file/')
def return_file():
    list_directory = tools.list_dir('restored_file')
    filename = './restored_file/' + list_directory[0]
    print ("**************************************")
    print (list_directory[0])
    print ("**************************************")
    # return send_file(filename, attachment_filename=list_directory[0], as_attachment=True)
    return send_file(filename, as_attachment=True)
    # return send_file(filename)
```

Fig. 3.16: Route to download the decrypted file

This route returns the encrypted file to the user. It finds the encrypted file in the restored_file directory and transmits it as a downloadable attachment.

- Route for home page

```
@app.route('/home')
def back_home():
    tools.empty_folder('key')
    tools.empty_folder('restored_file')
    return render_template('index.html')
```

Fig. 3.17: Route for home page

This route clears the key and restored_file folders before redirecting the user to the homepage (index.html).

- Route for data

```python
# after submitting on upload
@app.route('/data', methods=['GET', 'POST'])
def upload_file():
    tools.empty_folder('uploads')
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit a empty part without filename
        if file.filename == '':
            flash('No selected file')
            return 'NO FILE SELECTED'
        if file:
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], file.filename))
            return start_encryption()
        return 'Invalid File Format !'
```

Fig. 3.18: Route for handling data

This route manages the file upload feature. It accepts a POST request with a file attachment, saves it to the upload's directory, and starts the encryption process.

- Route to download keys

```python
@app.route('/download_data', methods=['GET', 'POST'])
def upload_key():
    tools.empty_folder('key')
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit a empty part without filename
        if file.filename == '':
            flash('No selected file')
            return 'NO FILE SELECTED'
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            # file.save(os.path.join(app.config['UPLOAD_KEY'], file.filename))
            file.save(os.path.join(app.config['UPLOAD_KEY'], "KEYS.pem"))
            return start_decryption()
        return 'Invalid File Format !'
```

Fig. 3.19: Route to download Keys

This channel is used to upload encryption keys. It receives a POST request containing an encryption key file, stores it to the key directory, and starts the decryption process.

3.10 KEY CHALLENGES

1) SECURITY CONCERNS

- Data Encryption: Developing a strong encryption scheme for ensuring safety of all user information that will be transmitted and stored.
- Access Control: Defining strong access, control policy to ensure that only authorized personnel can gain access into sensitive files.
- Secure Authentication: Maintaining robust security protocols including multi-factor authentication for the purpose of preventing unauthorized logins.

2) SCALABILITY

- Elastic Architecture: Scalable system design using cloud elasticity.
- Load Balancing: The process of load balancing whereby user requests are equally distributed among several servers.
- Database Scaling: Using scalable database decisions to manage the growing amount of data.

3) COLLABORATION FEATURES

- Real-Time Sync: Mechanisms/Systems of real time synchronization of the work files of multiple collaborators.
- Version Control: Include the option of creating version control capabilities that would follow users' tracks and handle all types of modifications they bring about.

4) COST MANAGEMENT

- Usage Analytics: Utilizing analytical tools to track user behavior and maximize resources deployment.
- Cost-Effective Storage: Using flexible storage technologies with pricing per use or number of reads.

5) INTEGRATION WITH CLOUD PROVIDERS

- API Compatibility: Periodic verification of API updates by cloud service providers, ensuring smooth integration.
- Data Migration: Creating effective approaches to transferring user's data across various cloud platforms.

6) USER EXPERIENCE

- Intuitive UI/UX: Performing user testing as well as feedback sessions, which is a necessary step for improving the UI quality and making it as much user friendly as possible.
- Performance Optimization: Improving efficiency of file uploads and downloads for speediness and quick responses.

7) REGULATORY COMPLIANCE

- Legal Expertise: Working together with legal experts to ascertain adherence of the information protection regulations or laws.
- Data Governance: Data governance policy, involving measures for compliance with laws on data processing, storage, and protection.

8) PERFORMANCE OPTIMIZATION

- Caching Mechanisms: The use of caching methodologies to boost efficiency and improve delay times.
- Code Optimization: Continuous assessment of the code base to enhance system performance.

9) BACKUP AND RECOVERY

- Incremental Backups: Adopting incremental backups to minimize data transfers and storage demands.
- Automated Recovery: Creating self-healing processes to reduce instances of downtime due to data loss.

10) CROSS-PLATFORM COMPATIBILITY

- Responsive Design: Testing for responsiveness and compatibility of the user interface on different kinds of screens and resolution.
- API Standardization: Implementing standardized APIs for compatibility purposes of varied OSs/devices.

CHAPTER 4: TESTING

4.1 TESTING STRATEGY

The reliability, security and functioning of the cloud-based secure FLS depend on testing. The testing strategy encompasses various types of testing to cover different aspects of the system -:

1) UNIT TESTING

- Objective: Check the consistency of its constituents and modules.
- Implementation: Unit tests are performed by developers for specific components of the code to check that every function acts in a proper manner.
- Tools: Unit testing is performed using unit testing frameworks, JUnit for Java and pytest for Python.

2) SYSTEM TESTING

- Objective: Compute the overall system behavior for the specified requirements.
- Implementation: Simulate real users' behavior in end-to-end testing covering function like file upload, download, access control, and encryption.
- Tools: Selenium for web application testing as well as JMeter for performance testing.

3) USER ACCEPTANCE TESTING (UAT)

- Objective: Make sure that the system satisfies user expectations and business requirements.
- Implementation: Testing is performed by real users, or stakeholders, to confirm whether this system meets their needs and requirements.
- Tools: Tools that manage UAT such as TestRail and PractiTest.

4) CROSS-BROWSER AND CROSS-PLATFORM TESTING

- Objective: Ensure that the system works properly through different browsers and systems.
- Implementation: Run it through Chrome, Mozilla, Safari, Windows, Macintosh or even Linux.
- Tools: BrowserStack, CrossBrowserTesting or Sauce Labs.

5) DATA BACKUP AND RECOVERY TESTING

- Objective: Ensure that data backup and recovery mechanisms work well.
- Implementation: Simulate scenarios of data loss and test the performance of backup recovery.
- Tools: Custom scripts or specialized tools for the selected backup and recovery software.

6) SCALABILITY TESTING

- Objective: Ensure that the system can accommodate additional load and data volume.
- Implementation: Ensure that the system can be capable of handling new more numbers of users, data files and transaction volume.

It is important to conduct continuous testing during the lifecycle to ensure that defects are detected and rectified beforehand, minimize project costs and produce an efficient and safe cloud-based system.

4.2 TEST CASES AND OUTCOMES

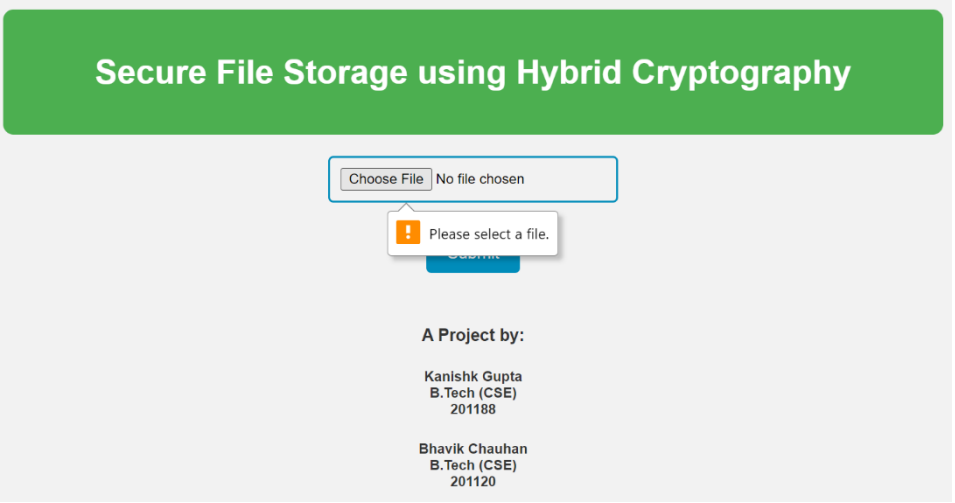


Fig. 4.1: Error message is shown when you click on Submit without choosing a file

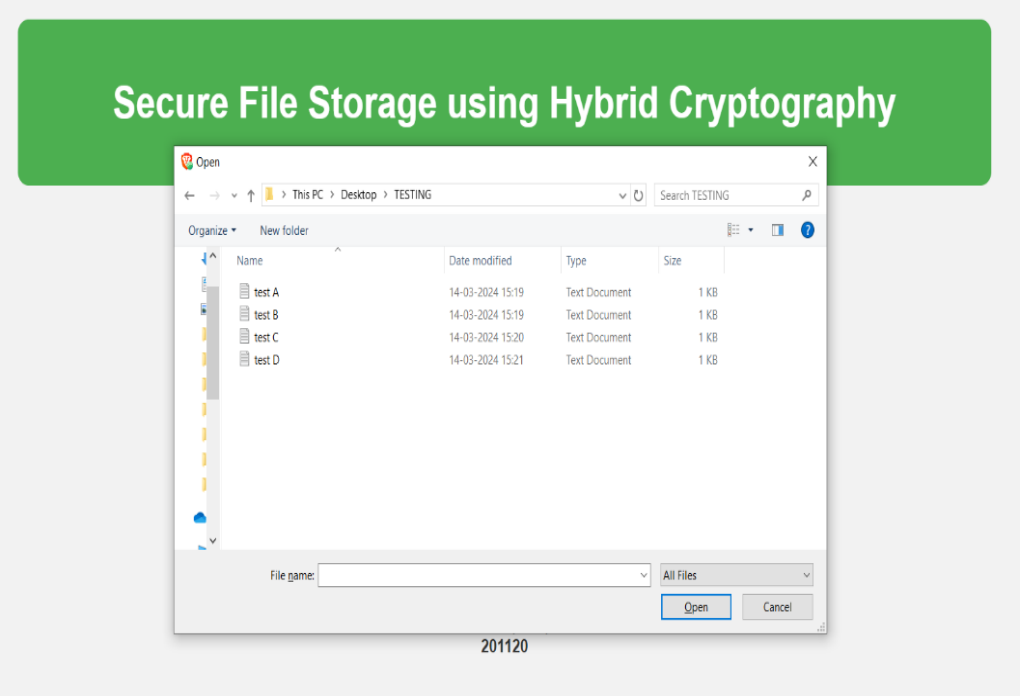


Fig. 4.2: Uploading a file

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

The web interface of the project, which has been hosted on the cloud like Render, is a great accomplishment because it makes the application available to the users over the internet. Cloud hosting services provide a platform where websites can be hosted, managed and accessed remotely, hence, scalability, reliability and easy deployment are possible. By the participation of the web-interface on Render, the team ensured that the users can use the application from anywhere with an internet connection without the necessity of local installation or configuration.

Fig. 5.1: Main page (when our code is compiled)

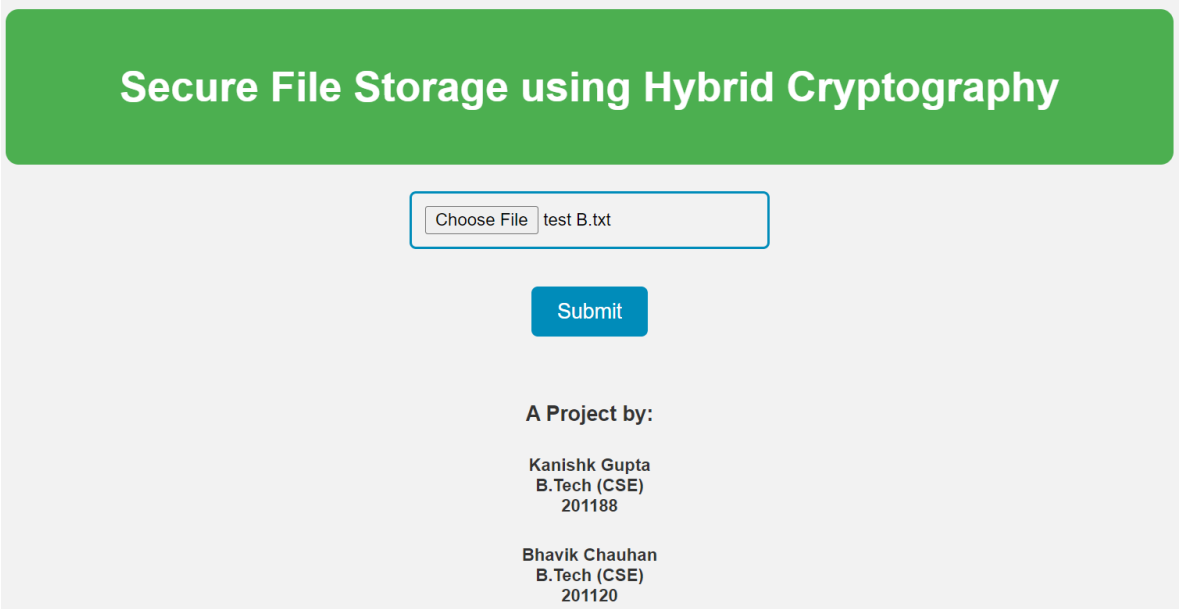


Fig. 5.2: Chosen a file to upload

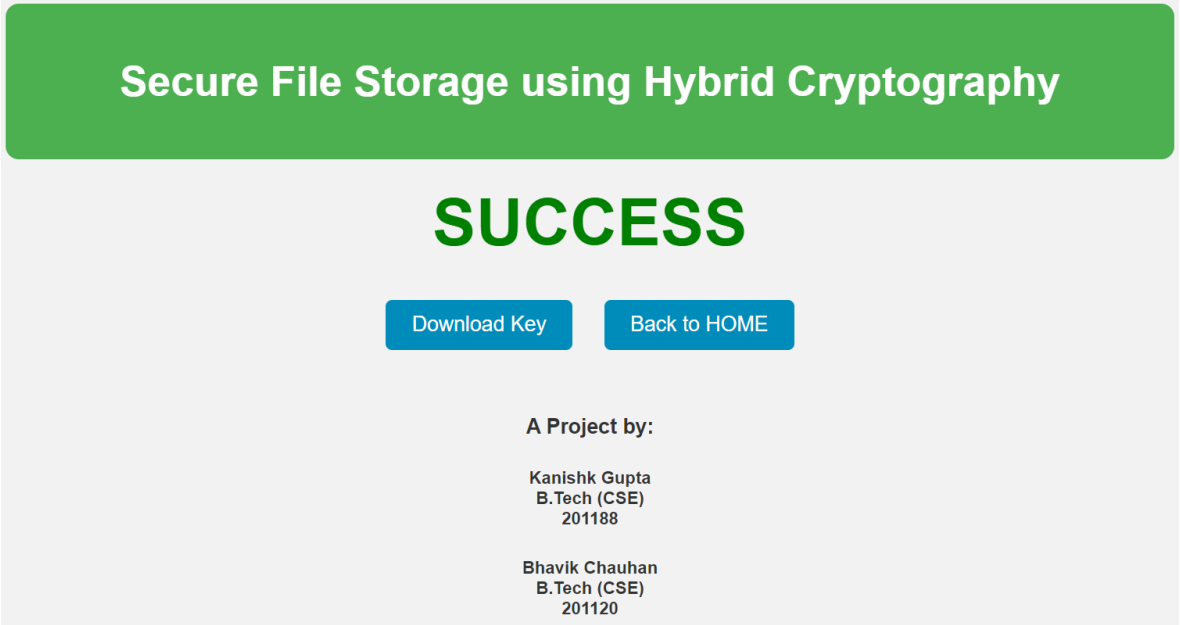


Fig. 5.3: To download the encrypted key

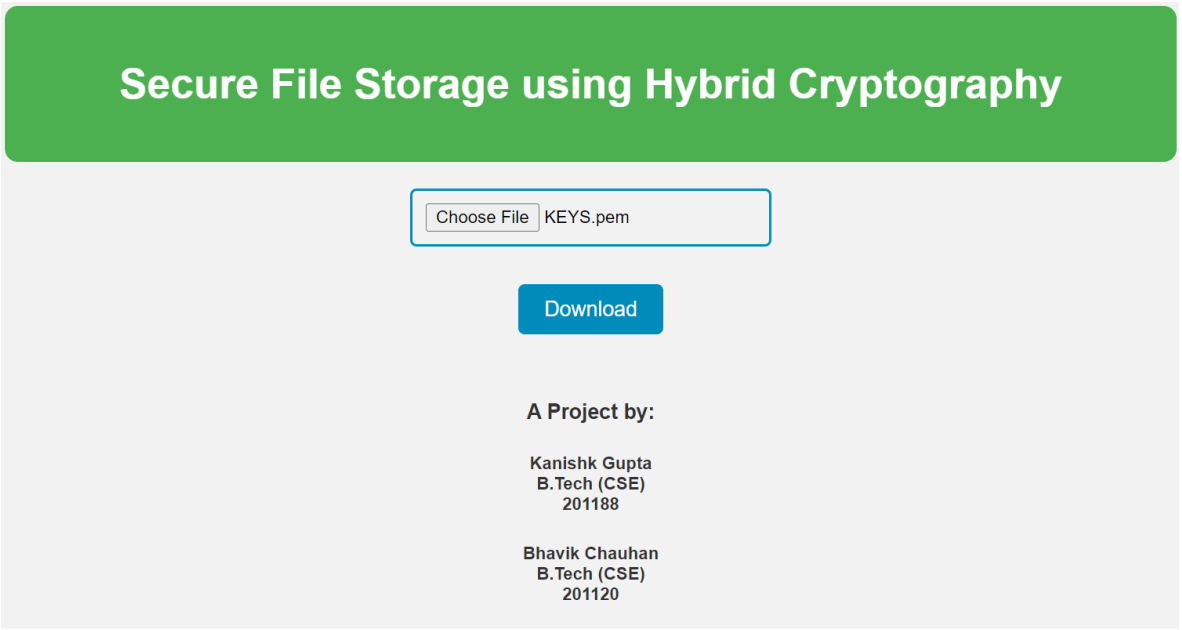


Fig. 5.4: Chosen an encrypted key to upload

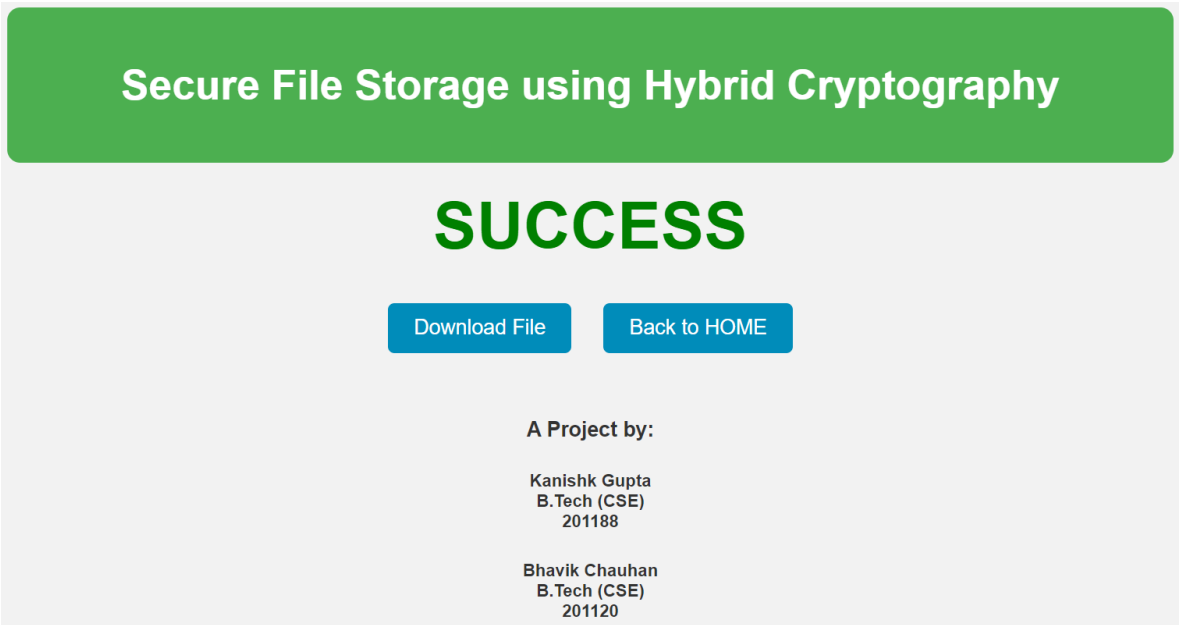


Fig. 5.5: To download the original file

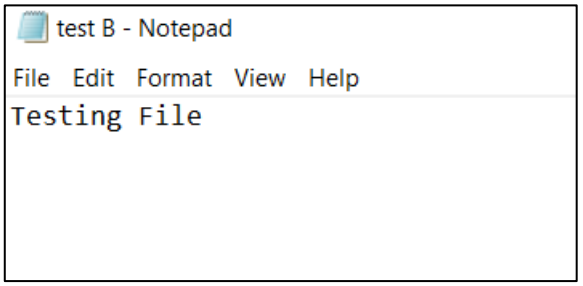


Fig. 5.6: Original File

5.2 COMPARISON WITH EXISTING SOLUTIONS

- **Security Approach:**

  Our Solution:

  Utilizes a hybrid cryptographic algorithm combining Fernet, ChaCha20Poly1305, AES-GCM (AES for encryption and Galois/Counter Mode (GCM) for authentication) and AES-CCM (AES encryption with Counter with CBC-MAC (CCM) authentication).

  Google Drive:

  Uses industry-standard encryption methods, including AES for data at rest.

  Provides server-side encryption, ensuring data security during transmission and storage.

  Offers two-factor authentication (2FA) for user accounts.

  Google Drive's security is well-established and audited regularly.

- **Key Management:**

  Our Solution:

  Stores encrypted keys

  Ensures that keys are not directly stored alongside the data.

  Google Drive:

  Manages encryption keys centrally.

  Provides robust key management infrastructure.

  Encrypts data using unique keys per file.

  Offers seamless key rotation and management.

- **User Experience:**

  Our Solution:

  Requires users to recover keys for the file for decryption.

  Google Drive:

  User-friendly interface.

  Simple file upload, download, and sharing.

  Transparent encryption; users don't need to manage keys manually.

- **Performance:**

  Our Solution:

  Claims efficiency in a cloud infrastructure.

  Performance may vary based on file size and encryption overhead.

Google Drive:

Optimized for speed and responsiveness.

Efficient data synchronization across devices.

Google's global infrastructure ensures low latency.

- **Data Availability and Redundancy:**

  Our Solution:

  Data availability depends on the cloud infrastructure.

  May lack features like automatic data replication.

  Google Drive:

  High availability with data replication across multiple data centers.

  Redundancy ensures data durability.

  Reliable service uptime.

- **Collaboration and Sharing:**

  Our Solution:

  Lacks advanced collaboration features.

  May not support real-time collaboration.

  Google Drive:

  Allows real-time collaboration on documents, spreadsheets, and presentations.

  Easy sharing with permissions control.

  Integrates seamlessly with Google Workspace (formerly G Suite).

- **Cost and Scalability:**

  Our Solution:

  Cost depends on the cloud provider and infrastructure.

  Scalability may be limited by design.

  Google Drive:

  Offers free storage with paid plans for additional space.

  Scales seamlessly based on user needs.

  Ideal for personal use and small to large organizations.

- **Compliance and Auditing:**

  Our Solution:

  Compliance may vary based on cloud provider.

Google Drive:

Complies with industry standards (e.g., GDPR, HIPAA).

Regularly audited for security and privacy.

Provides audit logs for administrators.

CHAPTER 6: CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSION

The key components of the user-friendly interface for uploading files and utilizing the advanced encryption methods make the cloud-based secure file storage system more efficient in terms of data security. Through this integration, the users can upload any type of the files for instance, documents, images, and others. This makes the system more accessible and convenient, and at the same time, it has stringent security measures. After uploading file, they get encrypted using the advanced algorithms that ensure the security of your data and protection from cyber threats.

Users are also given the ability to download a security key which can help safeguard the privacy and security of their data. With that key users may rest assure that their files remain impenetrable because no record of the original file or the key is kept by the system. These strategies by not only ensuring security of data but also give power to the user by allowing them to manage their digital information with confidence and satisfaction.

On the top of that, the addition makes the process of downloading and uploading files easy, allowing to restore files and return them to their original state with the minimum effort. This gives the users the peace of mind that they can safely restore their files knowing that their data is safe and that no one can interfere with the encryption and decryption process. Such a user-driven strategy to data security builds confidence within the users as well as an emblem of the system's willingness to protect their digital assets.

In short, the integration of an intuitive user interface for file upload and encryption, plus exceptional key management features, can be seen as a big step in the field of cloud security. When usability and accessibility are top priorities, together with stronger encryption practices, this system puts the users in control of safeguarding their private data without compromising on the convenience. Through our ongoing innovations in cloud security, a series of these kinds of solutions enables individuals and organizations to fearlessly use digital environment.

6.1.1 KEY FINDINGS

1) Implementation of hybrid encryption techniques

Using hybrid encryption techniques in addition to standard ways has become a crucial approach to enhancing data security and guaranteeing peak system performance. Through the integration of several encryption techniques, the system was able to achieve an increased degree of security against unapproved access and data breaches. This hybrid technique reduced possible weaknesses seen in single encryption approaches while strengthening the system's security posture. Furthermore, the hybrid encryption algorithms were implemented in a way that maintained the responsiveness and efficiency of the system, providing a smooth user experience without sacrificing performance metrics.

2) DNA-based bio-inspired cryptography

The use of DNA-based bio-inspired cryptography was one of the project's novel approaches, and it showed promise as a supplementary method to more traditional cryptography methods. This unique cryptography technique uses DNA molecules' intrinsic features to securely encode and decode digital information, taking inspiration from biological systems. The investigation of DNA-based encryption offers novel benefits, including intrinsic resistance to cyberattacks and possible scalability for handling massive amounts of data, opening new avenues for cryptographic study. Although it is still in its early phases, the incorporation of DNA-based encryption has enormous potential to improve cloud-based systems' security environment in the future.

3) Client-side encryption with TEEs

The use of client-side encryption, particularly TEEs such as Intel SGX, has emerged as a critical technique for improving data privacy and integrity in the cloud storage system. TEEs provide an additional degree of security against unwanted access and manipulation by moving the encryption and decryption procedures to the client side within a secure enclave. This client-side encryption solution gives customers more control over their data while reducing the dangers associated with server-side vulnerabilities and unwanted access by other parties. The usage of TEEs, such as Intel SGX, demonstrates a proactive approach to protecting sensitive data in cloud settings, creating trust and confidence in users about the confidentiality and integrity of their information.

4) Exploration of hybrid key generation approaches

The research investigated hybrid key generation methodologies, with the goal of improving the security and efficacy of cryptographic activities in the system. The system attempted to strengthen its encryption processes against potential flaws and security threats by merging several key generation procedures, including traditional cryptographic algorithms and novel DNA-based techniques. This hybrid method to key generation not only strengthens cryptographic operations, but also promotes flexibility to changing cybersecurity concerns. Furthermore, the combination of hybrid key generation methodologies with DNA-based cryptographic systems is a forward-thinking strategy for assuring the long-term security and viability of cloud-based services.

5) Comparison of hybrid cryptography schemes

A complete examination and comparison of several hybrid crypto schemes was a critical component of the project, allowing for the evaluation of their unique strengths, limitations, and potential for integration with cloud services. The project's goal was to discover the most effective and pragmatic solution to data security within the cloud storage system by testing several hybrid encryption algorithms and protocols. This comparative research aided informed decision-making in the selection of cryptographic algorithms that strike a balance between strong security protections and operational efficiency. Furthermore, the insights gained from comparing hybrid crypto schemes guided strategic decisions regarding algorithm selection, optimization, and customization, ensuring alignment with the cloud-based secure file storage system's specific security requirements and performance objectives.

6.1.2 LIMITATIONS

1) Theoretical and Conceptual Approach

The project was mostly based on theoretical and conceptual frameworks to create and assess the solutions, but this way the project may not have been able to fit into the real-world conditions. Theoretical models and conceptual frameworks, although, they are useful for guiding the research, are the abstraction of the real environment, hence, they do not cover the complex issues and the nuances in the real life. Thus, the theoretical framework of the project might have overlooked the practical aspects or problems that could be a barrier to the success and the feasibility of the solutions. To illustrate, the models that are developed theoretically may not be able to describe the details of the system interoperability, user behavior and the environmental factors that can affect the performance of the cybersecurity

measures. In addition, the theoretical approaches may be too simplistic to deal with the complicated problems of the real-world cybersecurity threats and vulnerabilities, which in turn could lead to some loopholes in the design or implementation of the solutions. Therefore, the theoretical frameworks are very important since they are the basis for research, but still, the application of these frameworks should be supported by the empirical testing and validation so that they can be meaningful in the real world.

2) Feasibility Concerns and Practical Considerations

One of the project's primary limitations is related to the viability and practicality of the suggested solutions, specifically regarding the cost and accessibility of the required hardware and infrastructure parts. For example, even while certain cryptography solutions show promise in theory, there may be obstacles to their actual use due to lack of funding, outdated technology, or financial concerns. The implementation of hardware security modules (HSMs) and other client-side devices may need significant expenditures in terms of money and technological know-how. Furthermore, especially in contexts with limited resources, the scalability of suggested solutions could be restricted by the accessibility of appropriate hardware or infrastructure. As a result, even while theoretical models could point to promising directions for improving cybersecurity, their actual use requires thorough assessment of practical limitations and the creation of scalable, affordable options that satisfy operational or organizational needs.

3) Need for further testing and validation

The preparedness and maturity of certain suggested solutions, especially those based on cutting-edge technology like bio-inspired cryptography, is another drawback. Although these technologies have the potential to completely transform cybersecurity procedures, their practical use necessitates extensive testing and validation against operational situations and real-world threats. Bio-inspired cryptography systems, for instance, could show flaws or performance constraints that require thorough testing procedures to find and fix. Furthermore, unless such solutions are put through testing and empirical validation, it is unclear how effective they will be in reducing modern cybersecurity risks. Consequently, even while the project's investigation of novel cryptographic techniques is praiseworthy, the iterative nature of cybersecurity innovation is highlighted by the necessity for more study, testing, and validation. The initiative establishes the foundation for future research

endeavours that strive to enhance and optimize nascent cybersecurity solutions for pragmatic implementation, by recognizing the significance of empirical validation.

6.1.3 CONTRIBUTIONS TO THE FIELD

1) The project is trying to enhance the data security in cloud-based environments by studying hybrid encryption techniques, DNA based cryptography and client-side encryption via the TEE which is the innovative solution to the problems. The project, by having its practical application of these techniques, offers a real insight into the workings, the pluses and the possible problems. For instance, the research on hybrid encryption shows the point that the mixture of symmetric and asymmetric encryption methods can raise the security while at the same time the performance will not be affected. Another good thing about the research of the DNA-based cryptography is the discovery of the benefits of the biomimicry for the creation of the strong encryption algorithms. Moreover, the research on the client-side encryption via TEE gives the explanation of the working mechanism and the security problems of the task of encryption transfer to the trusted hardware environment. Thus, the project enables stakeholders to learn how to put these cutting-edge cryptographic techniques into practice that in turn gives them the required insights to make the cloud-based file storage systems more secure.
2) The project's conclusions are of a great import to a lot of different people, including developers, system administrators, and organizations that are searching for a way to boost the security of their cloud-file storage systems. The project provides the practices that are realistic and the results which are based on the researches which assist the stakeholders to make the proper decisions about the security measures that they should implement. Designers can utilize the project's results to devise and include the strong encryption systems in the cloud-based applications thus, strengthening the data protection. Likewise, system administrators can apply the outcome of the project to the security settings and protocols of their cloud infrastructures, and thus, the weak points and threats will be eliminated. Besides, the organizations can use the project's ideas to the security plans and thus, the firstly, the sensitive data and the secondly, the regulation compliance will be improved. Towards the end, the project's importance to the stakeholders is that it can reveal how the scientific concepts can be transformed into the strategies of the cloud's security.
3) The project's dedication to promoting innovation and expanding the frontiers of traditional cryptography techniques is demonstrated by its investigation of novel

approaches including DNA-based cryptography and the development of keys inspired by biological processes. The research creates opportunities for better and more efficient encryption techniques that utilize ideas inspired by nature by breaking new ground. For example, research into DNA-based cryptography deviates from conventional cryptographic paradigms and provides an insight into the possibilities of biologically inspired encryption systems. In a similar vein, the investigation of biologically inspired key creation presents new methods for producing cryptographic keys that are based on biological processes, improving the robustness and variety of encryption techniques. The study establishes the foundation for future research initiatives targeted at expanding the boundaries of cybersecurity and encryption technology by adopting these novel approaches.

4) The project's focus on security issues in cloud-based applications emphasizes the significance of giving cybersecurity priority in the digital age. The initiative draws attention to the vital role that security plays in cloud computing settings and emphasizes the necessity of strong data protection protocols and proactive risk mitigation techniques. Moreover, the project stimulates continuous efforts to address new threats and vulnerabilities in online file storage systems by providing a foundation for further study and investigation in the field of cloud security. Through cultivating a security-aware and innovative culture, the project adds to the joint effort to protect confidential information and maintain confidence in cloud-based technology.

6.2 APPLICATIONS

Developing an internet system that is secure, which can also meet the demands of all the businesses which operate in the industries specified below offers to you a one-stop solution which is very handy and can be used by anyone. It is all about that signifies the presence of various safety attributes which have been conveniently designed to enable storage of real cash, financial statements, as well as private documents. At the same time the system is performing high-level data security and document management processes streamline and collaboration team members are also covered by it. The system helps to keep on track while industry regulations are maintained. For instance, the financial system, insurance companies, information technology services industries, and manufacturing have this system implemented to assure safe storage of confidential data, data consistency, and computer crime reduction.

Academic institutions or think tanks can establish platforms for researchers to exchange ideas and collaborate. All the information provided is 100% safe and proper platforms are produced to work out research data and intellectual assets. Such distributed system has got its data centrally maintained and so there are stringent access controls which in turn will boost research collaborations hence producing genuine research results while the privacy of researchers' results is still protected. Above that, an encryption of the information and an inbox audit of the system ensure the confidentiality and the incorruptibility of the research data that help researchers to separate themselves and practitioners in the academic circles with the aim of disseminating the scientific knowledge faster.

Two types of legal organizations that are critical for the functionality of the system are law firms as well as corporate legal entities (e. g. , regulators). These are inclusive of the maintenance of legal records, contracts, as well as other documents that are exposure-prone in a confidential way. Storing documents in encrypted and access-controlled systems not only guarantees the reliability but also is the main principle of legal documents change; while the audit trail of such system provides unchanged documents along with documents modifications validation proofs. Alongside the partnership of attorneys that this could bring, it may also incentivize organizations to evaluate whether they are following any data privacy laws or if they are compliant to any legal standards. Hence, the very implementation strengthens security and provides for enhanced control of access to documents; yet at the same time there arise other difficulties such as the possibility of data leakage and disclosure of information without one's consent.

The cloud-based secure file storage system is the healthcare sector platform that is trusted to store and manage patient records, sensitive health information, and compliance-related documents securely since the sector is in charge of the patient privacy and the data security. The doctors and the healthcare professionals utilize the encryption and access control features of the system to ensure that the patient data is kept secret and safe, while the audit trails assist to track the activities on the medical records and also to maintain the compliance with the healthcare regulations. Furthermore, the system is both scalable and reliable and a very good choice for managing the large amounts of electronic health records and medical imaging data created by the modern healthcare systems.

Educational institutions, like schools, colleges, and universities, use the system to manage the academic resources, research papers, and student records securely. The system gives

document management to teachers, researchers, and students and at the same time, the faculty members, researchers, and students can collaborate easily. Besides, the system guarantees that the confidential educational data is kept safe and private. Apart from the educational institutions, the system also aids them in the organization of the administrative processes, the establishment of the online learning facilities and the distance education system which thereby creates a secure and accessible learning environment for students and educators.

Government offices and defense agencies use the system to handle the classified documents, sensitive information, and national security assets securely. By employing the system's encryption, access control, and audit trail capabilities, the government entities can protect the sensitive data from the unauthorized access, insider threats, and cybersecurity attacks. Moreover, the system is scalable and resilient, thus, it is ideal for the management of large amounts of classified data and the support of mission-critical operations. Besides the government agencies and defense organizations, the system is also used for the collaboration, document workflows, and the compliance with the security and the regulatory requirements.

Banks, investment firms, and insurance companies use the system to secure financial transactions, client information, and regulatory documents. The system's encryption capabilities assure the confidentiality and integrity of financial data, and its access control features restrict the access to sensitive information based on user roles and permissions. Furthermore, the system's audit trails provide a verifiable record of financial transactions and document access, hence, the organizations can prove the compliance with industry regulations and internal policies. The system of financial data protection from unauthorized access and data breaches helps the financial institutions to preserve the customer trust, maintain their reputation, and reduce the financial risks.

The system provides an individual with a safe environment regarding the protection of personal data, confidential documents, and private files. The users can easily keep and manage their personal documents, such as legal documents, finance records, and private correspondence, and they are sure that their data is encrypted and protected from unauthorized people. The encryption of the system makes sure that the data is protected and kept away from the public, even if there is a data leak or a hacker who has no authorization. Via the system, people are taught to guard their privacy which results in privacy, data security and digital trust in a world that is more and more interconnected.

The system being a teamwork oriented is a great model for the modern workplaces that work on the team, the creativity and innovation. In places where employees work together, they can go online, edit, and share documents with each other, and then later they can work together across teams, departments, and locations. The version control of the system allows the teams to work with the latest version of the documents, and at the same time access control features enable the organizations to set up user roles and permissions according to job responsibilities and project requirements. Through the teamwork and, at the same time, securing the data privacy and safety of the system, the organizations can achieve a high level of productivity, effectiveness, and competitiveness which is a must in today's fast-paced business environment.

It can also be used in managing blockchain networks as they rely on cryptographic keys for secure transactions. Hybrid cryptographic system can manage and protect these keys, preventing unauthorized access to wallets and smart contracts. Enhanced security in blockchain applications contributes to the overall integrity of decentralized systems.

As the Internet of Things (IoT) grows, data generated by connected devices needs secure storage. Our system can handle encrypted sensor data, device logs, and telemetry information. The hybrid approach ensures that data from smart homes, industrial sensors, and wearable devices remains confidential.

It can be used to develop a secure voting platform where encrypted ballots are stored in the cloud. Voters can cast their votes securely, knowing that their choices are confidential. The hybrid approach prevents unauthorized access or manipulation of voting data.

In the fields such as media and entertainment, where the management of large volumes of digital assets is a must, the system can be considered as a secure storage for the media files, the video and the audio content and the creative assets. Content providers, producers, and media experts can safely store and manage their digital belongings, thus making sure that the important media files are properly preserved.

6.3 FUTURE SCOPE

1. **Integration with Emerging Technologies:** As technology continues to evolve, your system can explore integration with emerging technologies such as blockchain, machine learning, and quantum cryptography. Blockchain can enhance data integrity and transparency, while machine learning can improve threat detection and anomaly detection capabilities. Quantum cryptography offers potential advancements in encryption and key distribution, further strengthening the security of the system.

2. **Enhanced Privacy Features:** Future iterations of the system can focus on enhancing privacy features, such as zero-knowledge proofs and differential privacy techniques. Zero-knowledge proofs allow users to prove possession of certain information without revealing the information itself, offering greater privacy guarantees. Differential privacy ensures that statistical queries on the data do not reveal sensitive information about individual users.

3. **Advanced Access Control Mechanisms:** Future versions of the system can introduce advanced access control mechanisms, such as attribute-based access control (ABAC) and role-based access control (RBAC) with dynamic authorization. ABAC allows for fine-grained access control based on user attributes, while RBAC simplifies access management by assigning permissions based on predefined roles. Dynamic authorization enables real-time adjustments to access permissions based on changing user attributes or contextual factors.

4. **Enhanced User Experience and Usability:** Continuous refinement of the user interface and user experience design can further enhance the usability and accessibility of the system. This includes intuitive navigation, contextual help features, and personalized user dashboards, ensuring that users can easily navigate and utilize the system's features to meet their specific needs and preferences.

5. **Homomorphic Encryption:** Explore homomorphic encryption for secure computation on encrypted data. Users could perform operations (such as search or analytics) directly on encrypted files. This would preserve privacy while allowing useful computations.

REFERENCES

[1] B. R. Rao and B. Sujatha, "A hybrid elliptic curve cryptography (HECC) technique for fast encryption of data for public cloud security," in International Journal of Engineering and Technology, vol. 10, no. 4, pp. 186-188, 2018.

[2] R. Benadjila, L. Khati, and D. Vergnaud, "Secure storage—Confidentiality and authentication," Comput. Sci. Rev., vol. 44, p. 100465, 2022, doi: 10.1016/j.cosrev.2022.100465.

[3] M. da Rocha, D. C. G. Valadares, A. Perkusich, K. C. Gorgonio, R. T. Pagno, and N. C. Will, "Secure Cloud Storage with Client-Side Encryption Using a Trusted Execution Environment," in Proceedings of the 10th International Conference on Cloud Computing and Services Science CLOSER - Volume 1, pp. 31-43, 2020.

[4] M. I. Reddy, A. P. S. Kumar, and K. S. Reddy, "A secured cryptographic system based on DNA and a hybrid key generation approach," Biosystems, vol. 197, p. 104207, 2020, doi: 10.1016/j.biosystems.2020.104207.

[5] S. A. Ahmad and A. B. Garko, "Hybrid Cryptography Algorithms in Cloud Computing: A Review," 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2019, pp. 1-6, doi: 10.1109/ICECCO48375.2019.9043254.

[6] P. V. Maitri and A. Verma, "Secure file storage in cloud computing using hybrid cryptography algorithm," 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 2016, pp. 1635-1638, doi: 10.1109/WiSPNET.2016.7566416.

[7] S. Nepal, C. Friedrich, L. Henry and S. Chen, "A Secure Storage Service in the Hybrid Cloud," 2011 Fourth IEEE International Conference on Utility and Cloud Computing, Melbourne, VIC, Australia, 2011, pp. 334-335, doi: 10.1109/UCC.2011.55.

[8] M. Li, S. Yu, N. Cao and W. Lou, "Authorized Private Keyword Search over Encrypted Data in Cloud Computing," 2011 31st International Conference on Distributed Computing Systems, Minneapolis, MN, USA, 2011, pp. 383-392, doi: 10.1109/ICDCS.2011.55.

[9] S. Prasanth, "Cryptography Algorithms," Medium, [Online].

[10] "AWS Key Management Service (KMS) in a Nutshell," Medium, [Online].

[11] M. Batra, et al., "Secure file storage in cloud computing using hybrid encryption algorithm," International Journal of Computer Engineering Applications, vol. 19, no. 6, pp. 227-234, Jun. 2018

[12] M. Tahaseen et al, "Data Storage on Cloud Using Hybrid Encryption with One Time Password," IOSR Journal of Computer Engineering, vol. 19, no. 4, pp. 1-5, Aug. 2017.

[13] J. Thakur et al, "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis," International Journal of Emerging Technology and Advanced Engineering, vol. 1, no. 2, Dec. 2011.

[14] T. Jyoti et al, "Achieving Cloud Security Using Hybrid Cryptography Algorithm," International Journal of Advance Research and Innovative Ideas in Education, vol. 3, no.5, June 2017.

[15] A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, "Towards secure mobile cloud computing: A survey," Future Gen. Comput. Syst., vol. 29, no. 5, pp. 1278–1299, Jul. 2013.

[16] N. Singhal et al, "Comparative Analysis of AES and RC4 Algorithms for Better Utilization", International Journal of Computer Trends and Technology, Aug. 2011.

[17] G. Singh et al, "A Study of Encryption Algorithms (RSA, DES, 3DES and AES) for Information Security", International Journal of Computer Applications, vol. 67, no. 19, April 2013.

[18] Q. Zhang et al, "Cloud computing: state-of-the-art and research challenges," Journal of Internet Services and Applications, vol. 1, no. 1, pp. 7-8, 2010.

[19] M. Ali et al., "SeDaSC: Secure Data Sharing in Clouds," in IEEE Systems Journal, vol. 11, no. 2, pp. 395-404, June 2017.

[20] M. Armbrust et al, “A view of cloud computing,” Communications of the ACM, vol. 53, no. 4, p. 50, Jan. 2010.

[21] S. Joshi, “An efficient Paillier cryptographic technique for secure data storage on the cloud,” in 2020 IEEE 4th Int. Conf. on Intelligent Computing and Control Systems, Madurai, India, pp. 145–149, 2020.

[22] K. Benzekki, A. El Fergougui and E. A. Elbelrhiti, “A secure cloud computing architecture using homomorphic encryption,” International Journal of Advanced Computer Science and Applications, vol. 7, no. 2, pp. 293–298, 2016.

[23] A. Taha, D. S. Abd Elminaam and K. M. Hosny, “NHCA: Developing new hybrid cryptography algorithm for cloud computing environment,” International Journal of Advanced Computer Science and Applications, vol. 8, no. 11, pp. 479–486, 2017.

[24] D. Dasgupta, Z. Ji and F. Gonzalez, "Artificial immune system (AIS) research in the last five years," The 2003 Congress on Evolutionary Computation, 2003. CEC '03., Canberra, ACT, Australia, 2003, pp. 123-130 Vol.1, doi: 10.1109/CEC.2003.1299565.

[25] V. Masthanamma and G. L. Preya, “An efficient data security in cloud computing using the RSA encryption process algorithm,” International Journal of Innovative Research in Science, Engineering and Technology, vol. 4, no. 3, pp. 1441–1445, 2015.

[26] M. Storch, C. A. F. de Rose, "Cloud Storage Cost Modeling for Cryptographic File Systems," in Parallel, Distributed and Network-based Processing (PDP), 2017 25th Euromicro International Conference on, pp. 9-14, 2017, ISSN 2377-5750.

[27] Cloud security Alliance, “Security guidelines for critical areas of focus in cloud computing v3.0,” 2011.

[28] H. Tianfield, "Security issues in cloud computing," in IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, pp. 1082-1089.

[29] J. Kaur and S. Garg, "Security in Cloud Computing using Hybrid of Algorithms," International Journal of Engineering Research and General Science, vol. 3, no. 5, pp. [page range], Sep.-Oct. 2015.

[30] P. Shaikh and V. Kaul, "Enhanced Security Algorithm using Hybrid Encryption and ECC," IOSR Journal of Computer Engineering (IOSR-JCE), vol. 16, no. 3, pp. 80-85, May-June 2014.

[31] B. Kumar, J. Boaddh, and L. Mahawar, "A hybrid security approach based on AES and RSA for cloud data," International Journal of Advanced Technology and Engineering Exploration, vol. 3, no. 17, 2016.