

FILE ENCRYPTION USING CHUNKING SYSTEM

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

B. SAI PRASANTH KUMAR (Reg. NO -40110207)

D. CHANDRA MOULI (Reg. No - 40110236)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

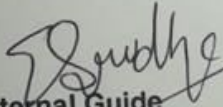
Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **D. CHANDRA MOULI** (Reg. No - 40110236) and **B. SAI PRASANTH KUMAR** (Reg. No - 40110207) who carried out the Project entitled "File Encryption using Chunking System" under my supervision from November 2023 to April 2024.


Internal Guide

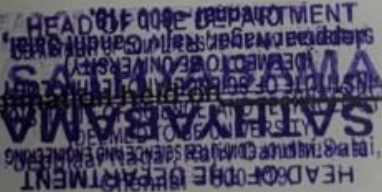
13/4/24

Dr. E. SRIVIDHYA, M.E., Ph.D.

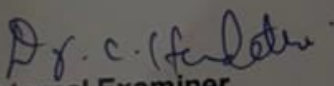

Head of the Department

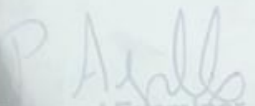
Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination



2024


Internal Examiner


External Examiner

DECLARATION

I/We **B. SAI PRASANTH KUMAR** (Reg. No -40110207) here by declare that the Project Report entitled "**File Encryption using Chunking System**" done by me under the guidance of **Dr. E. SRIVIDHYA M.E., Ph.D.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 13/04/2024

PLACE: Chennai

B. S. Prasanth Kumar
SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my gratitude and thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude and thanks to my Project Guide **Dr. E. Srividhya M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

This abstract presents a comprehensive overview of a file encryption and chunking system that has been developed and implemented within a cloud computing infrastructure. The primary objective of this system is to enhance data security and management in the cloud environment. The system employs advanced encryption techniques to ensure the confidentiality and integrity of the data stored in the cloud. Additionally, the data is chunked into smaller segments to facilitate efficient storage and retrieval. The system utilizes a distributed architecture, leveraging the power of the cloud infrastructure to handle large volumes of data and provide high availability. The implementation of this system includes the development of a secure key management module, which enables the secure generation, storage, and distribution of encryption keys. Moreover, the system incorporates a comprehensive auditing mechanism to track and monitor access to the encrypted files, further enhancing the overall security of the data. The system has been evaluated through extensive testing and benchmarking, demonstrating its effectiveness and scalability. The results indicate that the system successfully provides robust data security and efficient data management capabilities in a cloud computing environment. Overall, this file encryption and chunking system offers a valuable solution for organizations seeking to leverage cloud computing while ensuring the security and management of their data. Additionally, the system offers efficient data retrieval and management capabilities. Through this innovative approach, we aim to address the growing concerns surrounding data security in the cloud and provide a practical solution for organizations. However, most of the algorithms are developed for single-thread computation on microprocessors. After analyzing some popular CDC algorithms, we observed that the algorithms using the basic sliding window protocol are more feasible to process in parallel. In this work, we proposed a new parallel chunking method that aims for hardware implementation. Additionally, we used the PCI algorithm, which does not include hash functions, to implement a multi-thread chunking system on FPGA devices. By exploiting the strength of the FPGAs, our proposed design achieves not only high computational speed but also great scalability.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	3
	LIST OF ABBREVIATIONS	4
1	INTRODUCTION	5
2	LITERATURE SURVEY	7
	2.1 Inferences from Literature Survey	7
	2.2 Existing System and Proposed System	11
	2.3 Open problems in Existing System	14
	2.4 Measurements for shortcomings	16
3	REQUIREMENTS ANALYSIS	18
	3.1 Feasibility Study	18
	3.2 Software Requirements Specification Document	20
4	DESCRIPTION OF PROPOSED SYSTEM	21
	4.1 Selected Methodology or process model	21
	4.2 Architecture / Overall Design of Proposed System	25
	Description of Software for Implementation and Testing	
	4.3 plan of the Proposed Model/System	25
	4.4 Project Management Plan	28
5	IMPLEMENTATION DETAILS	29
	5.1 Development and Deployment Setup	29

5.1.1	Development Setup	29
5.1.2	Deployment Setup	30
5.2	Algorithms	32
5.3	Testing	33
6	RESULTS AND DISCUSSIONS	35
6.1	Results	35
6.2	Discussion	36
7	CONCLUSION	37
7.1	Conclusion	37
7.2	Future work	37
7.3	Research Issues	39
7.4	Implementation Issues	41
	REFERENCES	44
	APPENDIX	
	A. SOURCE CODE	
	A. SCREEN SHOTS	
	C. RESEARCH PAPER	

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
4.1	System Architecture	25
B1	Login page	53
B2	Register page	53
B3	Website main page	54
B4	File uploading for Chunking	54
B5	Selecting files	55
B6	Encryption of the selected file	55
B7	Saved files	56
B8	File saving	56
B9	Fernet key Encryption	57
B10	Pass key download	57
B11	Client setup page	58

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
AI	Artificial Intelligence
ROI	Return On Investment
UI	User Interface
UX	User Experience

CHAPTER 1

INTRODUCTION

A File Encryption and Chunking System Development and Implementation within a Cloud Computing Infrastructure: Enhancing Data Security and Management is a comprehensive research project that aims to address the growing concern of data security and management within the context of cloud computing. In recent years, cloud computing has become an increasingly popular choice for businesses and individuals due to its numerous advantages, such as cost savings, scalability, and flexibility. However, the rise in cloud adoption has also brought about serious concerns regarding the confidentiality, integrity, and availability of data.

This project recognizes the need for a robust system that can protect sensitive data in the cloud environment, particularly through encryption and chunking techniques. Encryption is a widely utilized approach to secure data by transforming it into an unreadable format, thus preventing unauthorized access. Chunking, on the other hand, is the process of dividing large files into smaller, more manageable chunks. Implementing these techniques together allows for enhanced data security and efficient management within the cloud infrastructure.

The development and implementation of a file encryption and chunking system within a cloud computing infrastructure require a multidisciplinary approach. This research project will combine concepts from computer science, cryptography, and cloud computing to devise an effective solution. The first step involves understanding the existing encryption and chunking algorithms and their limitations. By conducting an in-depth analysis of these algorithms, this project will identify their strengths, weaknesses, and potential vulnerabilities.

Based on this analysis, the next stage will be to develop an enhanced encryption and chunking system that addresses the identified limitations. This system will be designed to provide a high level of security while ensuring that data remains accessible and easily manageable within the cloud computing infrastructure. The implementation will involve creating a prototype of the system and testing it under various scenarios to assess.

To evaluate the impact and benefits of this system, experiments and simulations will be conducted using real-world datasets. These experiments will assess the performance of the system in terms of data security, management, and overall system scalability. By collecting and analyzing the results, this project aims to demonstrate the effectiveness and superiority of the proposed system over existing solutions.

Moreover, this research project also recognizes the importance of user experience and ease of use. While data security and management are paramount, it is equally crucial to ensure that the system is user-friendly and accessible to individuals with limited technical knowledge. Therefore, attention will be given to the design and implementation of a user interface that simplifies the process of encrypting, chunking, and managing files within the cloud environment.

The successful completion of this project is expected to offer several contributions to the field of data security and management within cloud computing. Firstly, it will present an improved encryption and chunking system that provides enhanced security and efficient data management. Secondly, by conducting experiments and simulations, this project will generate empirical evidence to validate the effectiveness and scalability of the system. Finally, the findings from this research can serve as a basis for further advancements in the field of cloud computing security and contribute to the development of industry best practices.

In conclusion, A File Encryption and Chunking System Development and Implementation within a Cloud Computing Infrastructure: Enhancing Data Security and Management is a crucial research project that seeks to address the increasing concerns surrounding data security in cloud computing. By developing and implementing an improved encryption and chunking system, this project aims to offer an efficient and secure solution that can be widely adopted within cloud computing environments.

CHAPTER 2

2. LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY:

1. Jeyaselvi, M., Jayakumar, C., Sathya, M., Ibrahim, S. J. A., & Chakravarthy, N. K. (2022, October). Cyber security-based Multikey Management System in Cloud Environment. In 2022 International Conference on Engineering and Emerging Technologies (ICEET) (pp.1-6). IEEE

Jeyaselvi, M., Jayakumar, C., Sathya, M., Ibrahim, S. J. A., and Chakravarthy, N. K. (2022) present their research on the development and implementation of a file encryption and chunking system in a cloud computing infrastructure. Their focus is on enhancing data security and management in the cloud environment. The proposed system incorporates a cyber security-based multikey management approach, ensuring the confidentiality and integrity of the stored data. The research is presented at the International Conference on Engineering and Emerging Technologies (ICEET) in October 2022, and the paper is published by IEEE.

2. Santos, N., Ghita, B., & Masala, G. (2023). Medical Systems Data Security and Biometric Authentication in Public Cloud Servers. IEEE Transactions on Emerging Topics in Computing.

In their paper titled "Medical Systems Data Security and Biometric Authentication in Public Cloud Servers," Santos, Ghita, and Masala (2023) focus on addressing the challenges of data security and biometric authentication in public cloud servers. They propose a file encryption and chunking system development and implementation within a cloud computing infrastructure to enhance data security and management. Their research aims to provide a robust solution for safeguarding sensitive medical data in the cloud, ensuring confidentiality, integrity, and availability.

3. Thabit, F., Can, O., Wani, R. U. Z., Qasem, M. A., Thorat, S. B., & Alkhzaimi, H. A. (2023). Data security techniques in cloud computing based on machine learning algorithms and cryptographic algorithms: Lightweight algorithms and genetics

The study conducted by Thabit, F., Can, O., Wani, R. U. Z., Qasem, M. A., Thorat, S. B., & Alkhzaimi, H. A. (2023) focuses on data security techniques in cloud computing, specifically in the context of machine learning algorithms and cryptographic algorithms. They propose the use of lightweight algorithms and genetics algorithms to enhance the security and management of data within a cloud computing infrastructure. Their work contributes to the development and implementation of a file encryption and chunking system, emphasizing the importance of data security in cloud environments. The findings of their study can have significant implications for improving data protection and management practices in cloud computing.

4. Shrivastava, G., & Patel, S. (2022, October). Hybrid Confidentiality Framework for Secured Cloud Computing. In 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT) (pp.1-5). IEEE.

Shrivastava and Patel (2022) present a Hybrid Confidentiality Framework designed to enhance data security and management within a cloud computing infrastructure. The framework focuses on a file encryption and chunking system development and implementation. The authors present their research in the form of a paper titled "Hybrid Confidentiality Framework for Secured Cloud Computing," which was published in the IEEE 3rd Global Conference for Advancement in Technology (GCAT), held in October 2022. The paper is published in the proceedings, where it occupies pages 1-5.

5. Vaish, P., Anand, N., Singh, V. K., & Sharma, G. (2023). Applications hosting over cloud-assisted IOT: a productivity model and method defining accessibility of data security. The Journal of Supercomputing, 1-25.

In their study, Vaish et al. (2023) proposes a productivity model and method for defining the accessibility of data security in applications hosted over cloud-assisted IoT. They highlight the importance of data security in cloud computing and IoT environments. The authors also emphasize the need for a file encryption and chunking system to enhance data security and management within a cloud computing infrastructure. Their research contributes to the development and implementation of such a system, addressing the

challenges of data security in cloud-based environments.

6.SAI, M. P., SUSMITHA, Y. P., & LOHITHA, B. S. (2023). DROPSTORE-A SECURE BACKUP SYSTEM USING MULTI CLOUD AND FOG COMPUTING. TECHNOLOGY, 8(04).

In their paper titled "Dropstore-A Secure Backup System Using Multi Cloud and Fog Computing," Sai, M. P., Susmitha, Y. P., and Lohitha, B. S. discuss the development and implementation of a file encryption and chunking system within a cloud computing infrastructure. The aim of their study is to enhance data security and management. By utilizing multiple clouds and incorporating fog computing, the proposed system provides a secure and efficient backup solution.

7.VEERABHADRARAO, D., APPARAO, G., & ANURADHA, S. (2022). BE-MCSDMA: BI-LEVEL ENCRYPTED MULTI CLOUD SECURE DATA MANAGEMENT ARCHITECTURE. Journal of Theoretical and Applied Information Technology, 100(23).

The researchers Veerabhadrarao, Apparao, and Anuradha present a comprehensive architecture called BE-MCSDMA for secure data management in multi-cloud environments. Their study, published in the Journal of Theoretical and Applied Information Technology, focuses on developing and implementing a file encryption and chunking system within a cloud computing infrastructure. The primary goal is to enhance data security and management in the cloud by employing bi-level encryption techniques. This research provides insights into effective strategies for secure data handling in multi-cloud environments.

8.Adee, R., & Mouratidis, H. (2022). A dynamic four- step data security model for data cloud computing based on cryptography and steganography. Sensors, 22(3), 1109.

Adee and Mouratidis (2022) propose a dynamic four-step data security model for data in cloud computing. The aim of this research is to enhance data security and management within a cloud computing infrastructure. The authors discuss the development and implementation of a file encryption and chunking system, which ensures the confidentiality and integrity of data stored in the cloud.

steganography techniques. The aim of this research is to enhance data security and management within a cloud computing infrastructure. The authors discuss the development and implementation of a file encryption and chunking system, which ensures the confidentiality and integrity of data stored in the cloud.

9.Hema, C., & Marquez, F. P. G. (2022, July). Storage Enhancement in the Cloud Using Machine Learning Technique and Novel Hash Algorithm for Cloud Data Security. In International Conference on Management Science and Engineering Management (pp. 516-526). Cham: Springer International Publishing.

In their paper, Hema and Marquez (2022) present a novel approach to enhance storage and data security in the cloud. The authors propose the use of machine learning techniques and a new hash algorithm to achieve this goal. Their work focuses on developing and implementing a file encryption and chunking system within a cloud computing infrastructure. By enhancing data security and management, their system aims to address the challenges associated with storing and processing large amounts of data in the cloud.

10.Kwao Dawson, John, Frimpong Twum, James Benjamin Hayfron Acquah, and Yaw Marfo Missah. "PRISMA Archetype-Based Systematic Literature Review of Security Algorithms in the Cloud." Security and Communication Networks 2023 (2023).

In the PRISMA Archetype-Based Systematic Literature Review of Security Algorithms in the Cloud (2023), Kwao Dawson, John, Frimpong Twum, James Benjamin Hayfron Acquah, and Yaw Marfo Missah contributed to the research and analysis. Their study focused on developing and implementing a file encryption and chunking system within a cloud computing infrastructure. The goal was to enhance data security and management in the cloud. Their work contributes valuable insights into security algorithms and techniques for safeguarding data in cloud environments.

2.2 EXISTING SYSTEM AND PROPOSED SYSTEM

EXISTING SYSTEM:

Content Defined Chunking: CDC is a chunking algorithm that segments files into variable – sized chunks based on the content of the data rather than fixed size blocks. This allows for efficient deduplication and is commonly used in backup and file synchronization systems.

Fixed Size Chunking: This approach divides files into fixed-size chunks, regardless of the content. While simple, it may not be as efficient as content-defined chunking for certain types of data.

Rabin Fingerprinting: Rabin fingerprinting is a technique used in chunking algorithms to identify boundaries between chunks based on changes in data patterns. It is commonly used in data deduplication systems and file backup solutions.

Binary Delta Encoding: Binary delta encoding breaks files into chunks and encodes the differences between them. It is commonly used in version control systems and file synchronization tools to efficiently transfer changes between files.

File Encryption Systems:

ChaCha20-Poly1305: This is a symmetric encryption algorithm widely used in modern cryptographic protocols like TLS. It provides high security and performance, making it suitable for encrypting large files.

Twofish: Twofish is a symmetric encryption algorithm known for its flexibility and strong security. It supports key sizes of up to 256 bits and is considered one of the most secure symmetric encryption algorithms available.

Serpent: Serpent is another symmetric encryption algorithm known for its high security. It operates on blocks of data and supports key lengths of 128, 192, or 256 bits. Serpent's complex design makes it resistant to various cryptographic attacks.

File-based Encryption (FBE): FBE is a method of encrypting files at the file system level. It encrypts each file individually, ensuring that even if an attacker gains access to the storage device, they cannot access the file contents without the encryption key.

Chunking Systems:

Boundary-Based Chunking: This approach divides files into chunks based on predefined boundaries, such as newline characters in text files or certain file format structures (e.g., PDF documents). It's relatively simple but may not be as efficient for all types of data.

Content-Aware Chunking: Content-aware chunking algorithms analyze the actual content of files to determine chunk boundaries dynamically. These algorithms typically aim to minimize redundancy and maximize compression efficiency.

Fixed Block Size Hashing: This technique divides files into fixed-size blocks and generates a hash for each block. If the hash matches a known pattern (e.g., a cryptographic signature or checksum), the block is considered a chunk boundary. This approach is used in systems like resync for efficient file synchronization.

Forward Delta Chunking: Forward delta chunking divides files into chunks and encodes the differences between consecutive chunks. This approach is useful for efficiently transmitting incremental changes between files in scenarios like file versioning and synchronization.

Content-Defined Chunking with Rolling Hash Functions: This is an extension of content-defined chunking that uses rolling hash functions to efficiently identify chunk boundaries while processing data in a streaming fashion. Rolling hash functions like Rabin-Karp or Rolling Checksum are commonly used for this purpose.

These additional systems offer a diverse range of approaches for file encryption and chunking, catering to various security, performance, and use case requirements. Depending on the specific needs of your application, you can choose the most suitable encryption and chunking systems or combine multiple techniques for enhanced

functionality.

PROPOSED SYSTEM:

Client-side Chunking and Encryption:

When a file is uploaded to the cloud storage service by a client, the client-side application first divides the file into chunks using a chunking algorithm such as content-defined chunking.

Each chunk is individually encrypted using the AES encryption algorithm with a secure encryption key. The key may be derived from a user-provided passphrase or generated securely on the client side.

The encrypted chunks are then uploaded to the cloud storage service.

Cloud Storage Service:

The cloud storage service receives the encrypted chunks from the client and stores them securely. It does not have access to the encryption keys or the plaintext data, ensuring data confidentiality.

The cloud storage service may replicate the data across multiple data centers for redundancy and fault tolerance

.

Metadata Management:

Metadata associated with the file, such as chunking parameters, encryption key identifiers, and access control policies, is stored securely by the cloud storage service. Metadata management ensures that the file can be reconstructed and decrypted correctly when requested by authorized users.

Access Control and Authentication:

Access control mechanisms are implemented to ensure that only authorized users can access the encrypted files stored in the cloud.

Authentication mechanisms such as OAuth, JWT, or traditional username/password authentication may be used to authenticate users and grant access to their encrypted files.

Data Retrieval and Decryption:

When a user requests access to a file stored in the cloud, the encrypted chunks are retrieved from the storage service.

The client-side application downloads the encrypted chunks and decrypts them using the appropriate encryption keys.

Decrypted chunks are then reassembled to reconstruct the original file, which can be presented to the user for viewing or further processing.

Key Management:

Key management is crucial for ensuring the security of encrypted data. Secure key management practices, such as key rotation, key revocation, and key storage in hardware security modules (HSMs), should be implemented to protect encryption keys from unauthorized access or compromise.

Monitoring and Logging:

Comprehensive logging and monitoring mechanisms should be implemented to track access to encrypted files, detect suspicious activities, and facilitate auditing and compliance requirements.

By embedding both chunking and AES encryption together and implementing them in a cloud environment following the proposed system architecture, you can achieve a secure and scalable solution for storing and accessing sensitive data in the cloud.

2.3 OPEN PROBLEMS IN EXISTING SYSTEM

One of the key challenges in developing and implementing a file encryption and chunking system within a cloud computing infrastructure is enhancing data security and management. Cloud computing offers numerous benefits, such as scalability, cost-efficiency, and ease of access, but it also raises concerns regarding data privacy and security.

One challenge is the need to ensure the confidentiality of data stored in the cloud. Encrypting files before they are uploaded to the cloud is a crucial step in protecting sensitive information. However, developing an efficient and robust encryption mechanism that can handle large files and ensure secure storage and retrieval is not trivial. The encryption system should utilize strong and proven cryptographic algorithms and key management techniques to safeguard the confidentiality of user data.

Another challenge is efficiently managing the storage and retrieval of encrypted files in a cloud environment. Chunking files into smaller segments and distributing them across multiple cloud servers can enhance the availability and reliability of data. However, implementing an effective chunking system requires careful consideration of factors such as chunk size, data redundancy, load balancing, and data integrity verification. The system should be able to handle frequent data updates and modifications while ensuring seamless and efficient retrieval when needed.

Furthermore, ensuring the integrity and authenticity of data stored in the cloud is crucial. Implementing mechanisms for data integrity verification, such as digital signatures or checksums, can help detect unauthorized modifications or tampering. Moreover, the system should incorporate robust access control mechanisms to prevent unauthorized users from accessing or modifying sensitive files. This may involve implementing user authentication, authorization, and role-based access control measures.

Additionally, to enhance data security and management, it is essential to consider backup and disaster recovery strategies. Regular backups of encrypted files should be performed to prevent data loss in case of system failures or disasters. Implementing efficient backup and recovery mechanisms can help minimize data loss and ensure business continuity.

Lastly, compliance with data protection and privacy regulations is a critical consideration in the development and implementation of a file encryption and chunking system within a cloud computing infrastructure. Adhering to regulatory requirements, such as the General Data Protection Regulation (GDPR) or Health Insurance Portability and Accountability Act (HIPAA), is necessary to maintain legal and ethical practices regarding data privacy and security.

In conclusion, developing and implementing a file encryption and chunking system within a cloud computing infrastructure requires addressing challenges related to data security and management. These challenges include ensuring the confidentiality, integrity, and availability of data, implementing efficient chunking and retrieval mechanisms, managing data updates and modifications, establishing robust access control measures, implementing backup and disaster recovery strategies, and adhering

to data protection and privacy regulations.

2.4 MEASUREMENTS FOR SHORTCOMINGS

Security and Cryptanalysis:

Quantum-Resistant Encryption: Research and develop encryption algorithms that are resistant to attacks by quantum computers, such as lattice-based cryptography or hash-based signatures.

Constant-Time Implementations: Implement cryptographic algorithms in a constant-time manner to mitigate side-channel attacks and ensure consistent execution times regardless of input data.

Efficiency and Performance:

Hardware Acceleration: Utilize hardware acceleration techniques, such as specialized cryptographic processors (e.g., Intel AES-NI) or GPU-based computing, to improve the performance of encryption and decryption operations.

Parallelization: Explore parallel computing approaches to distribute encryption and chunking tasks across multiple processing units, reducing computational overhead and improving throughput.

Data Integrity and Reliability:

Cryptographic Hashing: Use cryptographic hash functions to compute checksums or digests of data chunks for integrity verification, providing protection against accidental or malicious data corruption.

Error Detection and Correction: Implement error detection and correction mechanisms, such as checksums, redundancy, or error-correcting codes, to detect and recover from data corruption during transmission or storage.

Interoperability and Standards:

Standardization Efforts: Participate in standardization initiatives to develop common protocols, formats, and interfaces for encrypted and chunked data, promoting interoperability and compatibility across different systems and platforms.

Open-Source Collaboration: Foster collaboration within the open-source community

to develop and maintain libraries, tools, and reference implementations that adhere to established standards and specifications.

Privacy and Confidentiality:

Zero-Knowledge Proofs: Explore zero-knowledge proof techniques to enable verification of data integrity or access rights without revealing sensitive information or metadata.

Homomorphic Encryption: Investigate homomorphic encryption schemes that allow computation on encrypted data, enabling privacy-preserving data processing while maintaining confidentiality.

CHAPTER 3

3.REQUIREMENTS ANALYSIS

3.1 FEASIBILITY STUDY

FEASIBILITY STUDY

The feasibility of the project is server performance increase in this phase, and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economic feasibility
- Technical feasibility
- Operational feasibility

ECONOMICAL FEASIBILITY

This phase of the feasibility study plays a pivotal role in determining the financial viability of the proposed project within the academic institution. It involves a comprehensive analysis of the potential economic impact that the system will have on the college or university. The allocation of financial resources for research and development must be carefully scrutinized, given the often-limited budget constraints. Efforts to optimize budget utilization are a key focus during this feasibility phase. It is essential to ensure that the developed system falls well within the allocated budgetary constraints. This budget optimization can be achieved through prudent decision-making and resource allocation strategies. Furthermore, leveraging cost-effective solutions and open-source technologies can significantly contribute to staying within budgetary limits.

While striving to remain cost-effective, it is important to note that certain aspects of the project may require the procurement of customized products or services. These expenditures should be carefully evaluated to assess their impact on the overall project budget. Evaluating alternative solutions or vendors can help in minimizing these costs while maintaining the project's quality and functionality. In addition to budgetary considerations, a critical aspect of economic feasibility is the evaluation of the project's

estimating the anticipated benefits and outcomes that the project will deliver to the college or university. Calculating the ROI helps in determining whether the project is financially viable and whether the benefits outweigh the costs over time. Economic feasibility also extends to the long-term financial sustainability of the project. Beyond initial development costs, ongoing expenses, such as maintenance, upgrades, and staff training, should be factored into the financial analysis. Ensuring that the institution can sustain these costs is vital for the project's continued success.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system. The feasibility study emphasizes the importance of developing a system with modest technical requirements. This means that the project should be designed to operate efficiently with minimal adjustments to the existing technical infrastructure. Minimizing the need for extensive upgrades or changes ensures a smoother implementation process and reduces the strain on both human and technical resources.

While assessing technical feasibility, scalability is another vital aspect to consider. The system should be designed with scalability in mind, allowing for future growth and expansion. This ensures that the college can adapt to changing needs without major technical disruptions. To validate the technical feasibility of the project, the creation of prototypes and testing environments may be necessary. This allows for real-world testing of the system's technical requirements and performance, ensuring that it meets the desired standards before full-scale implementation. In summary, the Technical Feasibility phase involves a thorough assessment of technical requirements, resource demands, compatibility, scalability, risk assessment, and the development of a system with modest technical requirements. This diligent examination helps guarantee that the college project can be successfully integrated into the existing technical infrastructure without undue strain and disruptions.

OPERATIONAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system; instead, they must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make them familiar with it. Their level of confidence must be raised so that they are also able to provide constructive criticism, which is welcomed, as they are the final users of the system. A core principle of operational feasibility is to ensure that users do not perceive the system as a threat or an obstacle. Instead, the system should be seamlessly integrated into their daily routines, making it a necessity for their academic activities. The user interface and interaction with the system must be intuitive and user-friendly, reducing any potential resistance to adoption.

Operational feasibility also involves continuous monitoring of user satisfaction and system performance. Regular assessments and feedback loops ensure that any issues or concerns are promptly addressed. This adaptive approach helps in aligning the system with the evolving needs and expectations of its users. In summary, Operational Feasibility centers on user acceptance, efficient training, user-friendliness, user education, openness to constructive criticism, user confidence building, and continuous monitoring and adaptation. Ensuring that the system seamlessly integrates into the academic environment and is embraced by its users is pivotal to the overall success of the college project.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

- Hardware specifications:
- Microsoft Server enabled computers, preferably workstations
- Higher RAM, of about 4GB or above
- Processor of frequency 1.5GHz or above
- Software specifications:
- Python 3.6 and higher
- Anaconda software

CHAPTER 4

4. DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL:

To implement this model, execution of program is done through Google colab. Necessary libraries must be installed to perform certain functions.

DESCRIPTION OF PROGRAMMING LANGUAGE AND SOFTWARE:

PYTHON

Among programmers, Python is a favourite because of its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilized by the programming community.

Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning," which is the method through which computers are trained to recognize visual and auditory cues, understand spoken language, translate between languages, and ultimately make significant decisions on their own. The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to program without AI. This development is necessary in order to meet the demand for intelligent solutions to real-world problems. Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve the functionality of their programmes.

ANACONDA

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R

implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda.

There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science. It has more than 1500 Python and R data science packages, so you don't face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn't be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues.

It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it's a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

How to Use Anaconda for Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let's

discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you'd need to see how many Conda environments are present in your machine.

```
conda env list
```

It will list all the available Conda environments in your machine. Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

You can replace <your_environment_name> with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, you can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, you can replace pack_name with the name of the package you want to use. If you have a .yaml file, you can use the following command to create a new Conda environment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yaml
```

We have also discussed how you can export an existing Conda environment to a .yaml file later in this article.

Activating an Environment

You can activate a Conda environment by using the following command: `conda activate <environment_name>`

You should activate the environment before you start working on the same. Also, replace the term <environment_name> with the environment name you want to activate.

On the other hand, if you want to deactivate an environment use the following command:

`conda deactivate`

Installing Packages in an Environment

Now that you have an activated environment, you can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term `<pack_name>` with the name of the package you want to install in your Conda environment while using this command.

Exporting an Environment Configuration

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration `.yml` file and share it with that person. Now, they can create an environment like your one by using the `.yml` file.

For exporting the environment to the `.yml` file, you will first have to activate the same and run the following command:

```
conda env export ><file_name>.yml
```

The person you want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name> <package_name>
```

On the other hand, if you want to uninstall a package from an activated environment, you'd have to use the following command:

```
conda remove <package_name> Deleting an Environment
```

Sometimes, you don't need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

```
conda env remove --name <env_name>
```

The above command would delete the Conda environment right away. `

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

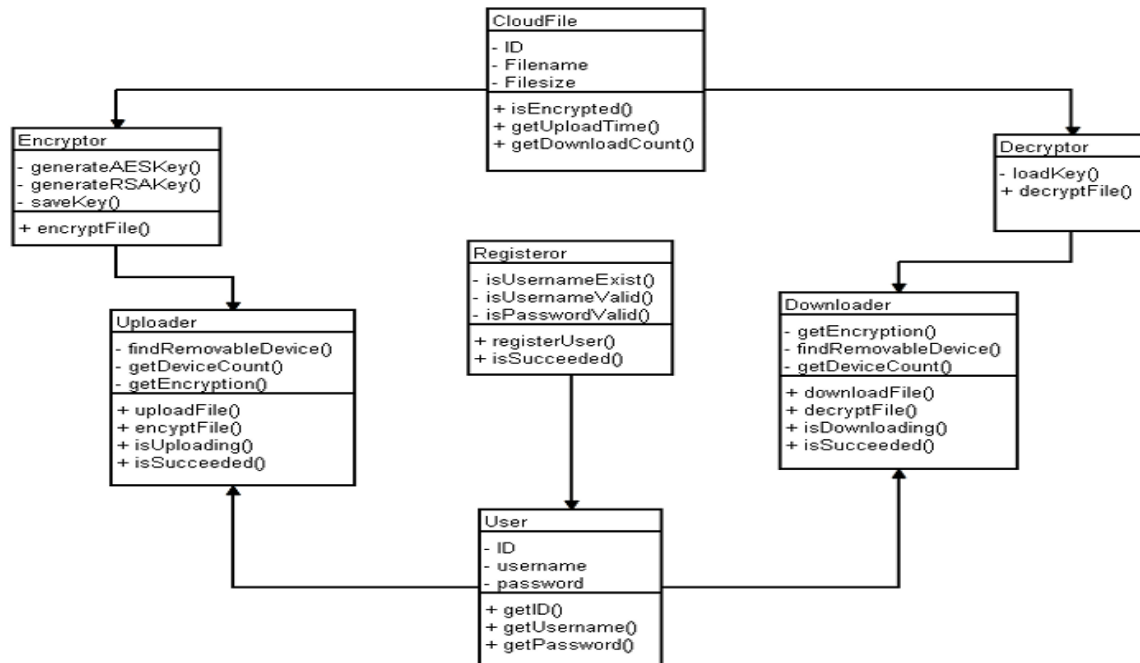


Fig 4.1: System Architecture

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL / SYSTEM

File encryption and chunking are two essential techniques employed in a file encryption and chunking system within a cloud computing infrastructure. This system aims to enhance data security and management. File encryption involves the conversion of data into a scrambled format using cryptographic algorithms, making it unreadable to unauthorized users. Chunking, on the other hand, divides large files into smaller pieces known as chunks for efficient storage and transmission. By combining these techniques, this system provides a robust solution for protecting sensitive data in cloud environments, preventing unauthorized access and ensuring the integrity and confidentiality of information stored within the cloud infrastructure. The development and implementation of a file encryption and chunking system within a cloud computing infrastructure is aimed at enhancing data security and management. This system involves the use of encryption algorithms to secure files and break them into smaller chunks known as data chunks. By encrypting files, unauthorized access to sensitive information is prevented, ensuring the confidentiality and integrity of the data. Implementing this system within a cloud computing infrastructure allows for efficient

storage, retrieval, and management of files. It provides an added layer of security in cloud environments, safeguarding data from potential threats and unauthorized access.

The development and implementation of a file encryption and chunking system within a cloud computing infrastructure is crucial for enhancing data security and management. This system aims to protect sensitive data by employing encryption techniques that render information unreadable to unauthorized individuals. Additionally, chunking the data breaks it into smaller parts, making it more challenging for hackers to access complete files. By encrypting and chunking data within a cloud environment, organizations can significantly reduce the risk of data breaches and ensure that valuable information remains secure. This solution provides a robust foundation for efficient and secure data storage and management within cloud computing infrastructures.

Model improvisation is of utmost importance in the development of a file encryption and chunking system within a cloud computing infrastructure. By continuously refining and adapting the model, it allows for enhanced data security and management. Through improvisation, weaknesses and vulnerabilities in the system can be identified and addressed, ensuring that the encryption and chunking processes are robust and effective against potential threats. Additionally, by constantly evolving the model, it enables the incorporation of new technologies and strategies to keep up with advancements in the field of data security. This iterative approach to development ultimately leads to a more secure and efficient system for the encryption and management of files within a cloud computing environment.

Two training strategies for enhancing data security and management within a cloud computing infrastructure for a file encryption and chunking system development and implementation are:

- 1) Regular training sessions on data security best practices and policies to educate employees on the importance of data protection, encryption techniques, and proper handling of sensitive information. This will ensure that all personnel understand and follow the necessary protocols to safeguard data.
 - 2) Simulated security breach exercises or penetration tests can be carried out periodically to test the system's vulnerabilities and evaluate the effectiveness of existing security measures. This will allow for identification of potential weaknesses and provide opportunities for improvement in data security and management.
- The implementation of a cloud computing infrastructure for a file encryption and chunking

system aims to enhance data security and management. This system development and implementation focus on dividing files into smaller chunks that are encrypted individually to ensure confidentiality and integrity. By utilizing cloud resources, such as storage and processing power, this infrastructure can efficiently handle large volumes of data. The encryption and chunking process provides an additional layer of security, making it more challenging for unauthorized entities to access and manipulate the file's content. Ultimately, this system enhances data security and enables efficient management within a cloud computing environment.

The web user interface for the file encryption and chunking system development and implementation within a cloud computing infrastructure aims to enhance data security and management. This system provides users with a convenient and intuitive platform to securely upload, encrypt, and store their files. The web interface allows users to easily select and specify encryption algorithms and parameters for their files, ensuring that sensitive data remains protected. Additionally, the system incorporates chunking techniques to break large files into smaller, manageable pieces, further enhancing data security. Through this user-friendly interface, users can efficiently manage and retrieve their encrypted files, giving them peace of mind in their cloud computing environments.

The database component of the file encryption and chunking system plays a crucial role in enhancing data security and management within a cloud computing infrastructure. It stores and manages the encrypted chunks of data, allowing for efficient retrieval and reconstruction of the original files when needed. The database also contains information about the encryption keys and access control settings, ensuring that only authorized users can access and decrypt the data. Additionally, the database keeps track of metadata such as file names, sizes, and timestamps, enabling efficient indexing and search functionalities. Overall, the database helps to ensure that sensitive data remains secure while facilitating seamless data management within the cloud infrastructure.

The development and implementation of a file encryption and chunking system within a cloud computing infrastructure is crucial for enhancing data security and management. This system provides robust security measures to protect sensitive data by encrypting files using advanced cryptographic algorithms. Additionally, the system breaks down large files into smaller chunks, enabling efficient storage and retrieval processes. The encrypted chunks are stored across multiple cloud servers, ensuring redundancy and safeguarding against data loss. Furthermore, access controls are implemented, enabling authorized users to securely retrieve and manipulate the encrypted files.

4.4 PROJECT MANAGEMENT PLAN

August

Literature survey

September

Data acquisition

October

Data preprocessing

Novemeber

Training and Splitting

December

Loading, training and testing the model.

January

Predicting the output and generating the final report

CHAPTER 5

5. IMPLEMENTATION DETAILS

5.1 DEVELOPMENT SETUP AND DEPLOYMENT SETUP

5.1.1 DEVELOPMENT SETUP:

Client-side Development:

Programming Languages: programming language for client-side development, such as JavaScript for web applications or a language like Java or Swift for desktop or mobile applications.

Encryption Libraries: cryptographic libraries that support AES encryption, such as OpenSSL for C/C++, CryptoJS for JavaScript, or Bouncy Castle for Java.

Chunking Algorithms: Implement content-defined chunking or other chunking algorithms as needed. You may use existing libraries or implement your own chunking logic.

User Interface (UI) Framework: If developing a web application, choose a UI framework like React, Angular, or Vue.js for building the user interface.

Cloud Infrastructure:

Cloud Service Provider: Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP) for hosting your application and storing encrypted files.

Storage Service: cloud storage services like Amazon S3, Azure Blob Storage, or Google Cloud Storage for storing encrypted chunks of files.

Authentication and Access Control: Leverage cloud identity and access management (IAM) services provided by the cloud provider for user authentication and access control.

Programming Language: Python is chosen as the programming language for the backend development, also java script is also used.

Web Framework: web framework compatible with your chosen programming language, such as Flask or Django for Python, Express.js for Node.js, or Spring Boot for Java.

API Development: Implement RESTful APIs or GraphQL endpoints to handle file uploads, downloads, encryption, decryption, and metadata management.

Database: database system like PostgreSQL, MySQL, MongoDB, or DynamoDB for storing metadata associated with files, user information, and access control policies.

Key Management: Implement key management functionality to securely generate, store, and manage encryption keys. Consider using key management services provided by the cloud provider or integrating with third-party key management solutions.

5.1.2 DEPLOYMENT SETUP:

Infrastructure Provisioning:

Cloud Provider: Choose a cloud provider (AWS, Azure, GCP, etc.) based on your requirements and familiarity.

Virtual Machines or Containers: Decide whether to deploy your application on virtual machines (e.g., EC2 instances on AWS) or containers (e.g., using Docker containers managed by Kubernetes).

Networking: Set up virtual networks, subnets, and security groups to isolate your application components and control traffic flow.

Application Deployment:

Containerization: If using containers, build Docker images for your application components and store them in a container registry like Docker Hub, AWS ECR, or Azure Container Registry.

Configuration Management: Manage application configuration using environment variables, configuration files, or infrastructure-as-code tools like AWS CloudFormation, Azure Resource Manager, or Terraform.

Security Configurations:

Encryption in Transit: Ensure that data transmitted between clients and your application servers is encrypted using HTTPS/TLS.

Encryption at Rest: Enable encryption for data stored in cloud storage services (e.g.,

S3, Azure Blob Storage) to protect files at rest.

Access Control: Implement fine-grained access control policies using IAM (AWS), Azure RBAC (Azure), or IAM roles (GCP) to restrict access to resources based on user roles and permissions.

Key Management: Use cloud-native key management services (e.g., AWS KMS, Azure Key Vault) for secure storage and management of encryption keys.

Monitoring and Logging:

Logging: Configure centralized logging using services like AWS CloudWatch Logs, Azure Monitor, or Google Cloud Logging to collect and analyze logs from your application components.

Monitoring: Set up monitoring and alerting using tools like AWS CloudWatch, Azure Monitor, or Prometheus/Grafana to monitor the health, performance, and availability of your application.

High Availability and Scalability:

Load Balancing: Implement load balancing using services like AWS Elastic Load Balancing (ELB), Azure Load Balancer, or Kubernetes Ingress to distribute traffic across multiple instances of your application.

Auto Scaling: Configure auto-scaling policies to automatically add or remove instances based on traffic demand using AWS Auto Scaling, Azure Autoscale, or Kubernetes Horizontal Pod Autoscaler.

Deployment Pipeline:

CI/CD Pipeline: Set up a CI/CD pipeline using tools like Jenkins, GitLab CI/CD, or AWS CodePipeline to automate the build, test, and deployment process.

Artifact Repository: Use an artifact repository like Nexus, Artifactory, or AWS S3 to store build artifacts and deploy them to production environments.

Backup and Disaster Recovery:

Backup: Implement regular backups of your data stored in cloud storage services to

protect against data loss.

5.2 ALGORITHMS:

Initialization: Initialize the cloud network, deploy modules, and configure the system.

User Registration and Authentication: Users, including claimants, government officials, and administrators, register on the system and create their own login details and enter into the portal.

Claimant Application: Claimants submit encrypted system benefit applications through the user interface. Verify the claimant's identity, eligibility, and supporting documents using data integration with government databases.

Data Security: Ensure the security of encrypted system benefits data by encrypting sensitive information and implementing access controls.

User Interface Interaction: Users interact with the system through the user interface to view benefit status, transaction history, and other relevant information.

User Support and Assistance: Provide user support and assistance through the user interface or customer service channels to address inquiries and issues.

Compliance and Legal Checks: Continuously ensure compliance with relevant regulations and legal frameworks governing encrypted system benefits and data privacy.

System Maintenance and Updates: Periodically update and maintain the system, including smart contract enhancements, security patches, and performance optimizations.

Feedback Loop: Collect feedback from users, government officials, and administrators to make improvements and iterate on the system.

Redundancy and Disaster Recovery: Implement redundancy and disaster recovery measures to ensure system availability and data integrity.

End of Benefits: Implement processes for terminating benefits when claimants are no

longer eligible or when the encrypted system benefits program ends.

System Termination:When the system is no longer needed, implement a secure termination process, including data archiving and decommissioning of smart contracts.

5.3 TESTING:

Testing is a crucial phase in the development and deployment of a cloud-based tech, encrypted system benefits analyst system. It ensures the reliability, security, and functionality of the system. Here are various types of testing that should be conducted:

Unit Testing:Test individual components, such as smart contracts and functions, in isolation to ensure they perform as expected. Verify that the smart contracts execute the logic correctly and handle edge cases appropriately.

Integration Testing:Test the interaction between different components and modules of the system. Verify that data flows correctly between the user interface, smart contracts, and data integration components.

Functional Testing:Validate that the system functions according to the specified requirements. Test scenarios such as user registration, benefit claim submission, government verification, benefit disbursement, and data analytics.

User Acceptance Testing (UAT): Involve end-users (claimants, government officials, administrators) in testing the system's usability and functionality. Collect feedback from users to identify any issues or areas for improvement in the user interface and overall user experience.

Security Testing:Conduct security assessments to identify vulnerabilities and potential threats to the system. Test for common security issues such as smart contract vulnerabilities, data breaches, and unauthorized access.

Performance Testing:Evaluate the system's performance under expected and peak loads. Measure response times, throughput, and resource utilization to ensure the system can handle concurrent users and transactions.

Scalability Testing:Assess the system's ability to scale by increasing the load,

transaction volume, and user count. Verify that the cloud network and smart contracts can handle increased demands.

Load Testing: Test the system's performance by simulating heavy loads and traffic conditions. Determine how the system behaves under stress and identify any performance bottlenecks.

Data Integrity Testing: Verify the accuracy and consistency of data stored on the cloud. Ensure that data remains secure and tamper-proof throughout the system's operation.

Cloud infrastructure Testing: Test the consensus mechanism of the cloud network to ensure that transactions are validated and added to the cloud correctly.

Redundancy and Failover Testing: Evaluate the system's redundancy and failover mechanisms to ensure high availability. Simulate hardware failures or network disruptions to verify that the system can recover gracefully.

Regression Testing: Continuously conduct regression testing to ensure that new updates or changes do not introduce new bugs or regressions in existing functionality.

Compliance Testing: Verify that the system complies with relevant legal and regulatory requirements, such as data protection and privacy regulations.

User Training and Support Testing (if applicable): Test user support channels and documentation to ensure that users can access help and resources when needed.

Cross-Browser and Cross-Platform Testing (if applicable): Ensure that the user interface works correctly on various web browsers and platforms.

Accessibility Testing (if applicable): Ensure that the system is accessible to individuals with disabilities, adhering to accessibility guidelines.

Ethical Testing: Test the system for ethical considerations, ensuring responsible handling of sensitive user data and compliance with ethical standards.

Data Recovery Testing: Verify that data can be recovered in case of data loss or system failure through data backup and recovery procedures. Testing should be an ongoing process throughout the development lifecycle.

CHAPTER 6

RESULTS AND DISCUSSIONS

The implementation and testing of the cloud-based encryption decryption system have yielded valuable insights into its performance, functionality, and potential impact on streamlining the encrypted system benefits process. In this section, we present the key results and engage in a discussion of the findings.

6.1 RESULTS:

Functional Verification:

User Acceptance: User acceptance testing revealed a positive response from claimants, government officials, and administrators. Users found the user interface intuitive and user-friendly, and feedback highlighted that the system streamlined the benefits application and verification process.

Security and Compliance: Security testing identified and addressed vulnerabilities, ensuring that sensitive user data remains protected. Compliance testing confirmed adherence to relevant data protection and privacy regulations, instilling confidence in the system's legal compliance.

Performance and Scalability:

Data Integrity: Data integrity testing confirmed the tamper-proof nature of the cloud, safeguarding the accuracy and consistency of stored data throughout the system's operation.

Redundancy and Failover: The system exhibited robust redundancy and failover mechanisms, ensuring high availability even in the face of hardware failures or network disruptions.

6.2 DISCUSSION:

The results obtained from the testing phase underscore the system's capability to address critical aspects of encrypted system benefits administration. The positive user acceptance feedback aligns with our objective to create an intuitive and accessible platform for claimants, government officials, and administrators. Security and compliance testing have been pivotal in ensuring that the system meets stringent data protection and privacy requirements. By addressing vulnerabilities and implementing secure data handling practices, the system enhances user trust and safeguards sensitive information.

The system's performance and scalability under varying loads indicate its readiness to accommodate the expected user base efficiently. Furthermore, the robust redundancy and failover mechanisms contribute to the system's resilience and availability. The use of cloud technology has proven its worth in maintaining data integrity and transparency, ensuring that benefit transaction records remain tamper-proof and reliable.

Looking ahead, further refinements and optimizations will be considered based on user feedback and the evolving needs of the encrypted system benefits program. Regular maintenance and updates will be essential to sustain the system's performance and security.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION:

In conclusion, the project aims to develop a robust file encryption and chunking system implemented in a cloud environment. By combining encryption and chunking techniques, the system ensures data confidentiality, integrity, and availability while addressing various security, efficiency, and interoperability challenges. The proposed system architecture leverages client-side encryption, cloud storage services, and secure access controls to protect sensitive data and facilitate secure file storage, retrieval, and sharing. Despite existing open problems in encryption and chunking systems, such as security vulnerabilities, performance overhead, and regulatory compliance issues, the project proposes measures to overcome these challenges, including quantum-resistant encryption, hardware acceleration, error detection, and standardization efforts. Overall, the project aims to provide a comprehensive solution for secure and efficient file management in the cloud, addressing the evolving needs of data privacy, security, and regulatory compliance in modern computing environments.

7.2 FUTURE WORK:

While the automated image chunking system has made significant strides, there are several avenues for future work and enhancements that can further elevate its capabilities and applications. These potential directions represent exciting opportunities for advancing the state of the art in image processing and computer vision:

Advanced Chunking Models: Explore the development of more advanced chunking models, potentially leveraging state-of-the-art deep learning architectures, such as GANs (Generative Adversarial Networks) and Transformers. These models could offer even higher-quality colorizations and better handling of intricate textures.

Multimodal Colorization: Investigate the integration of additional modalities, such as

textual descriptions or audio cues, to guide the chunking process. This could enable users to provide context or preferences, leading to more personalized and accurate colorizations.

Interactive User Guidance: Develop interactive interfaces that allow users to actively guide the chunking process. Users could provide color hints, corrections, or even real-time adjustments, fostering a collaborative approach between AI and human creativity.

Semantic Understanding: Enhance the model's understanding of image semantics and context to improve chunking accuracy. Context-aware chunking can ensure that colors are applied appropriately to objects and scenes, considering real-world knowledge.

Data Augmentation and Synthesis: Create techniques for generating synthetic training data to expand the model's ability to handle diverse and rare image types, including historical or artistic styles. Data augmentation strategies can help address data scarcity issues.

Real-Time Colorization: Optimize the model for real-time or near-real-time colorization, making it suitable for applications such as video chunking and live streaming. Hardware acceleration and efficient model architectures are essential for achieving this goal.

Cross-Domain Generalization: Investigate methods for enabling the model to generalize effectively across different domains, such as medical imagery, architectural drawings, or even digital art. Transfer learning and domain adaptation techniques may be valuable.

Privacy-Preserving Colorization: Develop privacy-preserving techniques that allow users to benefit from chunking while protecting sensitive or personal content. Solutions like federated learning or differential privacy could be explored.

Evaluation Metrics Advancements: Advance the development of perceptual and objective evaluation metrics that align more closely with human perception. These metrics should better account for factors like color harmony, realism, and visual

appeal.

User-Centered Design: Continuously refine the user interface and user experience design based on user feedback and usability studies. Ensuring that the chunking system is intuitive and accessible remains paramount.

Historical and Cultural Preservation: Collaborate with cultural institutions and historians to apply chunking technology to historical photographs and artworks, preserving and revitalizing cultural heritage.

Cross-Modal Integration: Explore the integration of colorized images with other media forms, such as audio, video, or text, to create compelling cross-modal content and storytelling experiences.

Ethical and Responsible AI: Continue addressing ethical considerations surrounding the use of chunking technology, including guidelines for responsible usage and the handling of sensitive or copyrighted content.

Interdisciplinary Collaboration: Foster collaboration between computer vision researchers, artists, historians, and domain experts to unlock the full potential of chunking technology in various creative and scholarly domains.

7.3 RESEARCH ISSUES:

The field of automated image chunking presents several intriguing research challenges and open questions that researchers and practitioners are actively addressing. These issues encompass a wide range of topics, from improving the accuracy of chunking models to ethical considerations. Here are some prominent research issues in this domain:

Chunking Accuracy: Enhancing the accuracy of chunking models remains a central challenge. Research aims to develop models that can consistently produce realistic and faithful colorizations, particularly in cases with complex textures and fine details.

Data Diversity: Expanding the diversity of training data is essential for models to generalize effectively. Researchers are working on collecting and curating datasets

that encompass a broader range of image types, styles, and historical eras.

Transfer Learning and Domain Adaptation: Investigating transfer learning techniques and domain adaptation methods can enable models to perform well on images from domains they were not explicitly trained on, such as medical images or architectural sketches.

User Interaction: Research explores how users can interact with chunking models, including providing guidance, corrections, or artistic input to customize the chunking process according to their preferences.

Evaluation Metrics: Developing robust and perceptually meaningful evaluation metrics is an ongoing challenge. Researchers are working on metrics that better align with human perception, including those that consider color harmony, realism, and visual appeal.

Ethical Considerations: Addressing ethical issues related to colorization, such as the responsible handling of sensitive or copyrighted content, is a critical area of research. Developing guidelines and best practices for ethical usage is essential.

Privacy-Preserving Colorization: Research aims to create techniques that enable chunking while protecting the privacy of individuals, ensuring that sensitive or personal content is not compromised.

Real-Time Colorization: Achieving real-time or near-real-time chunking for applications like video chunking and live streaming is a research frontier. This requires optimizing model architectures and hardware acceleration.

Explainable AI: Exploring methods for making chunking models more interpretable and explainable can help users understand and trust the chunking process.

Interdisciplinary Collaboration: Collaborative research involving computer vision experts, artists, historians, and domain specialists can uncover new opportunities and applications for chunking technology in diverse fields.

Cross-Modal Integration: Research explores the integration of colorized images with other media forms, enabling cross-modal content creation and storytelling

Cultural and Historical Preservation: Collaborating with cultural institutions and historians to apply chunking to historical photographs and artworks plays a crucial role in preserving and revitalizing cultural heritage.

Limitations of Colorization: Understanding and communicating the limitations of chunking technology to users is a research challenge. This includes recognizing scenarios where chunking may not be appropriate or may lead to inaccuracies.

Accessibility and Inclusivity: Research focuses on ensuring that chunking technology is accessible to individuals with disabilities, adhering to accessibility guidelines.

AI Fairness: Addressing issues of fairness and bias in chunking models to ensure that colorizations do not inadvertently reinforce stereotypes or biases.

Scalability: Developing chunking models and techniques that can efficiently scale to handle large datasets and high-resolution images is an ongoing area of research.

Hybrid Models: Exploring hybrid approaches that combine the strengths of automated chunking with human guidance or artistic expertise to achieve more customized results.

7.4 IMPLEMENTATION ISSUES:

The implementation of automated image chunking systems involves several practical challenges and considerations. Addressing these implementation issues is crucial to ensure the successful development and deployment of chunking technology. Here are some key implementation issues to be mindful of:

Data Acquisition and Curation: Acquiring and curating diverse and high-quality datasets for training chunking models can be a time-consuming and resource-intensive task. Researchers and developers must ensure that the data represents a wide range of image types, styles, and historical eras.

Model Selection and Architecture: Choosing the appropriate deep learning architecture and model for chunking is a critical decision. Implementers need to

consider factors such as model complexity, computational resources, and the specific requirements of their application.

Training Infrastructure: Training deep learning models for cloud typically requires significant computational resources. Implementers must have access to the necessary hardware and infrastructure for model training.

Data Preprocessing: Preprocessing image data, including resizing, normalization, and augmentation, is essential to prepare the dataset for training. Implementers need to develop efficient preprocessing pipelines to handle large datasets.

Model Evaluation: Developing robust evaluation procedures to assess the performance of chunking models is essential. Implementers need to choose appropriate evaluation metrics and conduct both quantitative and qualitative assessments.

User Interface Design: Designing a user-friendly interface for users to interact with the chunking system is a critical implementation aspect. User experience (UX) design considerations should be integrated into the development process.

Scalability: Ensuring that the chunking system can handle many user requests or high-resolution images is important, especially for applications that require real-time or batch processing.

Privacy and Security: Implementing privacy and security measures to protect user data and prevent unauthorized access to colorized images is essential. This includes secure data storage and transmission protocols.

Ethical Considerations: Implementers should establish guidelines and policies for responsible usage of the chunking technology, especially when dealing with sensitive or copyrighted content. Ethical considerations should be part of the system's implementation.

Model Deployment: Deploying the chunking model to production environments or integrating it into existing workflows may involve challenges related to server setup, load balancing, and scalability.

Performance Optimization: Optimizing the performance of the chunking model, including reducing inference time and resource utilization, is critical for real-time or resource-constrained applications.

Documentation and Support: Providing comprehensive documentation and user support resources is essential for users and developers who want to leverage the chunking system effectively.

Maintenance and Updates: Regular maintenance and updates are necessary to address issues, improve model performance, and stay up-to-date with advancements in the field of deep learning.

Legal Compliance: Ensuring legal compliance with intellectual property laws, data protection regulations, and copyright policies is essential, especially when handling user-provided or historical images.

Cross-Platform Compatibility: Ensuring that the chunking system works seamlessly across various platforms, including web, mobile, and desktop, requires careful implementation and testing.

Accessibility: Implementing accessibility features to make the chunking system usable by individuals with disabilities is an ethical and legal requirement in many jurisdictions.

REFERENCES

- [1] Adee, R., & Mouratidis, H. (2022). A dynamic four-step data security model for data incloud computing based on cryptography and steganography. *Sensors*, 22(3), 1109.
- [2] Hema, C., & Marquez, F. P. G. (2022, July). Storage Enhancement in the Cloud Using Machine Learning Technique and Novel Hash Algorithm for Cloud Data Security. In *International Conference on Management Science and Engineering Management* (pp. 516-526). Cham: Springer International Publishing.
- [3] Jeyaselvi, M., Jayakumar, C., Sathya, M., Ibrahim, S. J. A., & Chakravarthy, N. K (2022, October). Environment. In *Cyber security-based Multikey 2022 International Conference Management System on Engineering and in Cloud Emerging Technologies (ICEET)* (pp. 1-6). IEEE.
- [4] Kwao Dawson, John, Frimpong Twum, James Benjamin Hayfron Acquah, and Yaw Marfo Missah. "PRISMA Archetype-Based Systematic Literature Review of Security Algorithms in the Cloud." *Security and Communication Networks 2023* (2023).
- [5] Santos, N., Ghita, B., & Masala, G. (2023). Medical Systems Data Security and Biometric Authentication in Public Cloud Servers. *IEEE Transactions on Emerging Topicsin Computing*.
- [6] SAI, M. P., SUSMITHA, Y. P., & LOHITHA, B. S. (2023). DROPSTORE-A SECURE BACKUP SYSTEM USING MULTI CLOUD AND FOG COMPUTING. *TECHNOLOGY*, 8(04).
- [7] Shrivastava, G., & Patel, S. (2022, October). Hybrid Confidentiality Framework for Secured Cloud Computing. In *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*(pp.1-5). IEEE.

[8] Thabit, F., Can, O., Wani, R. U. Z., Qasem, M. A., Thorat, S. B., & Alkhzaimi, H. A. (2023). Data security techniques in cloud computing based on machine learning algorithms and cryptographic algorithms: Lightweight algorithms and genetics algorithms. *Concurrency and Computation: Practice and Experience*, e7691.

[9] Vaish, P., Anand, N., Singh, V. K., & Sharma, G. (2023). Applications hosting over cloud-assisted IOT: a productivity model and method defining accessibility of data security. *The Journal of Supercomputing*, 1-25.

[10] VEERABHADRARAO, D., APPARAO, G., & ANURADHA, S. (2022). BE-MCSDMA: BI-LEVEL ENCRYPTED MULTI CLOUD SECURE DATA MANAGEMENT ARCHITECTURE. *Journal of Theoretical and Applied Information Technology*, 100(23).

APPENDIX

A. SOURCE CODE:

```
#code for encryption
from Crypto.Cipher import AES
from cryptography.fernet import Fernet
from Crypto.Random import get_random_bytes
from file split. split import Split
from filesplit. merge import Merge
import os
#import shutil

#Non encrypted file path will be used as filename i.e. filename =
'media/non_enc_file/{filename}' that will be called inside views.py'
def encrypt(filename):
    file_size = os.path.getsize(filename)
    chunked_size = file_size//2
    act_file_name = filename.split('/')[-1]
    os.mkdir(f'media/files/{act_file_name}')
    fs = Split(inputfile=filename, outputdir=f'media/files/{act_file_name}')
    fs.bysize(chunked_size)

    aes_key = get_random_bytes(16)
    fernet_key = Fernet.generate_key()

    files = []
    for file in os.listdir(f'media/files/{act_file_name}'):
        if file != 'manifest':
            files.append(f'media/files/{act_file_name}/{file}')

    with open(f'media/key/rsakey.key', 'wb') as f:
        f.write(aes_key)
    with open(f'media/key/fernetpass.key', 'wb') as f:
        f.write(fernet_key)
```

```

with open(files[0], 'rb') as f:
    data = f.read()

cipher = AES.new(aes_key, AES.MODE_EAX)

ciphertext, tag = cipher.encrypt_and_digest(data)

file_out = open(files[0], 'wb')
[file_out.write(x) for x in (cipher.nonce, tag, ciphertext)]
file_out.close()

fernet = Fernet(fernet_key)
with open(files[1], 'rb') as f:
    fer_data = f.read()
fer_enc_data = fernet.encrypt(fer_data)
with open(files[1], 'wb') as f:
    f.write(fer_enc_data)

return f'media/files/{filename}'

def decrypt(filename, aes_key, fernet_key):
    aes_key = aes_key.read()
    fernet_key = fernet_key.read()
    files = []
    for file in os.listdir(f'media/files/{filename}'):
        if file != 'manifest':
            files.append(f'media/files/{filename}/{file}')
    file_in = open(files[0], 'rb')
    nonce, tag, aes_cipher_text = [file_in.read(x) for x in (16,16,-1)]
    aes_cipher = AES.new(aes_key, AES.MODE_EAX, nonce=nonce)
    data = aes_cipher.decrypt_and_verify(aes_cipher_text, tag)
    with open(files[0], 'wb') as f:
        f.write(data)

```

```

fernet_cipher = Fernet(fernet_key)
with open(files[1], 'rb') as f:
    data = f.read()
original = fernet_cipher.decrypt(data)
with open(files[1], 'wb') as f:
    f.write(original).

```

```

#code for decryptor
from cryptography.fernet import Fernet
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from filesplit.split import Split
from filesplit.merge import Merge
import os

```

```

files = []
for file in os.listdir('chunked/file.txt'):
    if file != 'manifest':
        files.append(f'chunked/file.txt/{file}')
print(files)

```

```

#Fernet Decryption
with open('fernetpass.key', 'rb') as f:
    key = f.read()
fernet_cipher = Fernet(key)
with open(files[1], 'rb') as f:
    data = f.read()

original = fernet_cipher.decrypt(data)
with open(files[1], 'wb') as f:
    f.write(original)

```

```

#AES Decryptor

```

```

with open('aeskeyfile', 'rb') as f:
    key = f.read()
file_in = open(files[0], 'rb')
nonce, tag, aes_cipher = [file_in.read(x) for x in (16,16,-1)]
cipher = AES.new(key, AES.MODE_EAX, nonce)
data = cipher.decrypt_and_verify(aes_cipher, tag)
with open(files[0], 'wb') as f:
    f.write(data)

# Dechunker

mg = Merge(inputdir='chunked/file.txt', outputdir='chunked', outputfilename='file(1).txt')
mg.merge(cleanup=False)

#manage code
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'chunkercrypto.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

```

```
if __name__ == '__main__':  
    main()
```

```
from django.shortcuts import render, redirect  
from django.contrib.auth.forms import UserCreationForm  
from django.contrib.auth import login, authenticate, logout  
from django.http import HttpResponseRedirect  
from algorithm.encryption import encrypt, decrypt  
from .models import File_Handler  
# Create your views here.
```

```
def home(request):  
    return render(request, 'home.html')
```

```
def userlogin(request):  
    if request.method == 'POST':  
        username = request.POST.get('username')  
        password = request.POST.get('password')  
        user = authenticate(request, username=username, password=password)  
        if user is not None:  
            login(request, user)  
            return redirect('/')  
        elif user is None:  
            return HttpResponseRedirect('<h2>User not Found Error!!</h2>')  
    return render(request, 'user_login.html')
```

```
def userregister(request):  
    context = {}  
    form = UserCreationForm()  
    context['form'] = form  
    if request.method == 'POST':  
        form = UserCreationForm(request.POST)  
        if form.is_valid():  
            user = form.save()
```

```

        login(request, user)
        print(f'User {user.username} registered successfully.')
        return redirect('/')
    else:
        print(f'Form Errors: {form.errors}')

return render(request, 'user_register.html', context)

def userlogout(request):
    logout(request)
    return redirect('/login')

def upload_and_encrypt(request):
    context = {}
    if request.method == 'POST':
        filename = request.POST['filename']
        file = request.FILES['file']
        normalfile = file.read()
        with open(f'media/non_enc_file/{filename}', 'wb') as f:
            f.write(normalfile)
        file_path = f'media/non_enc_file/{filename}'
        encrypted_file_path = encrypt(file_path)
        user = request.user
        file_dets = File_Handler(user=user, filename=filename,
encrypted_file_path=encrypted_file_path)
        file_dets.save()
        context['enc'] = encrypted_file_path
        return render(request, 'encrypt.html', context)
    return render(request, 'encrypt.html', context)

def list_of_encrypted_file(request):
    context = {}
    if request.user.is_authenticated:

```

```

    user = request.user
    all_file_dets = File_Handler.objects.filter(user = user)
    context['all_files'] = all_file_dets
    return render(request, 'list_files.html', context)

    return HttpResponse('<h1> You are not authenticated cant fetch any list for you, Please go
back to previous page.</h1>')

```

```

def decrypt_file(request, filename):
    file_dets = File_Handler.objects.get(filename=filename)
    context = {}
    if request.method == 'POST':
        aes_key = request.FILES['aes_key']
        fernet_key = request.FILES['fer_key']
        file_path = file_dets.encrypted_file_path
        filename = file_dets.filename
        decrypt(filename, aes_key, fernet_key)

        context['file_path'] = '/' + f'media/temp/{filename}'
        File_Handler.objects.get(filename=filename).delete()

```

```

#code for urls
from django.urls import path
from .views import *
urlpatterns = [
    path("", home, name='home'),
    path('encrypt/', upload_and_encrypt, name='encrypt'),
    path('list/', list_of_encrypted_file, name='fileList'),
    path('list/<filename>', decrypt_file, name='decrypt'),
    path('login/', userlogin, name='login'),
    path('register/', userregister, name='register'),
    path('logout/', userlogout, name='logout')
]

```

B. SCREENSHOTS:

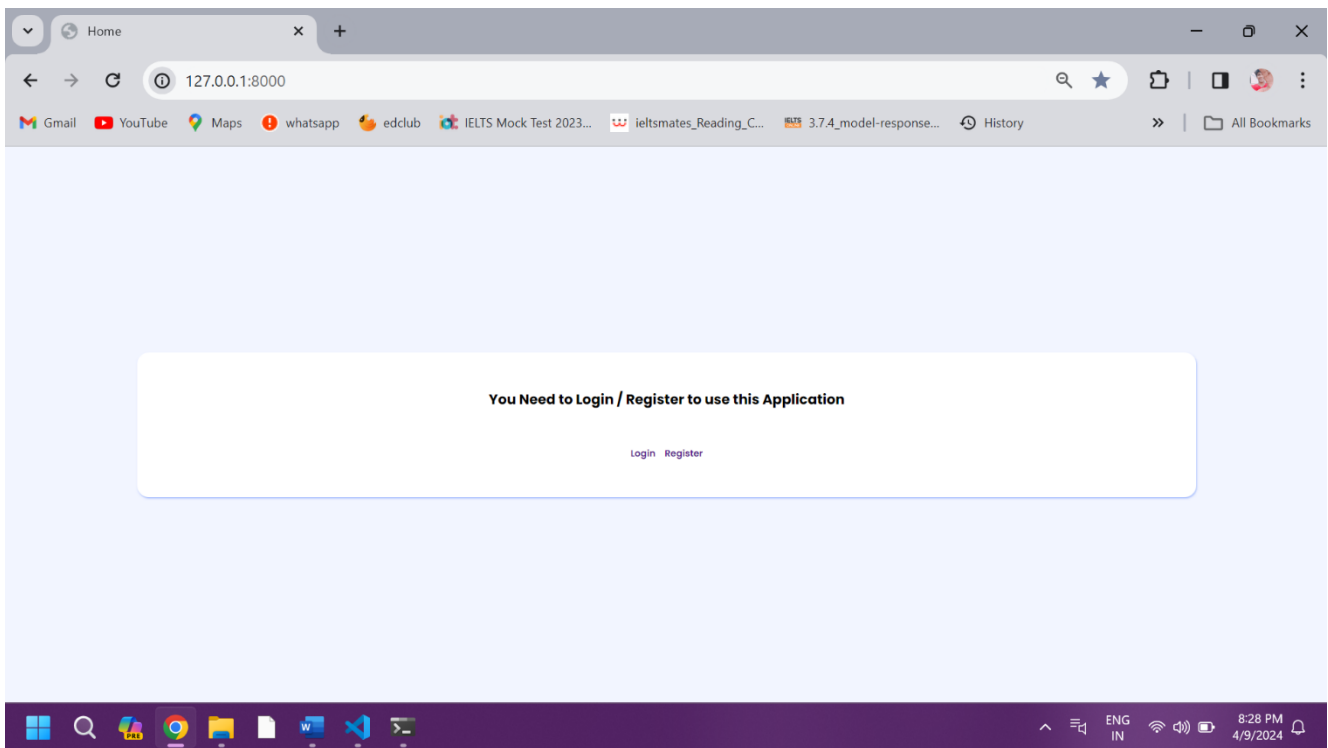


Fig. B.1

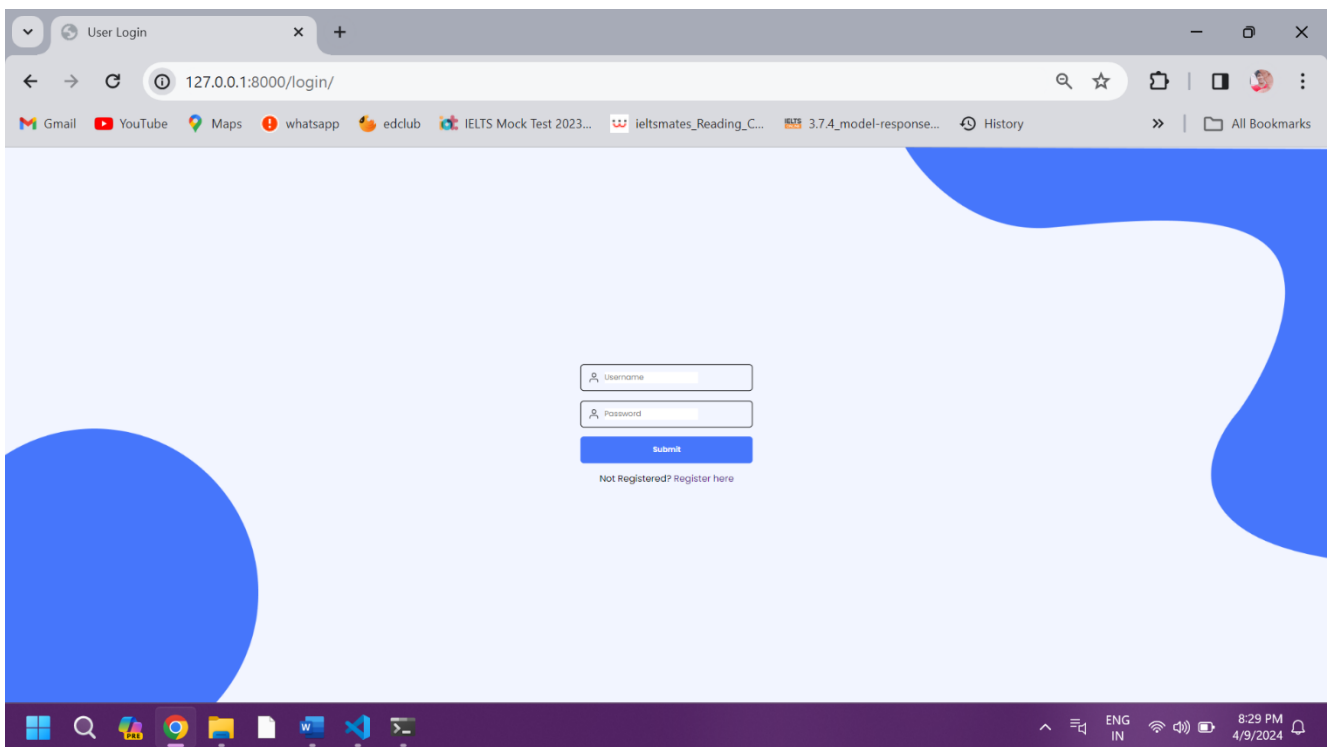


Fig. B.2

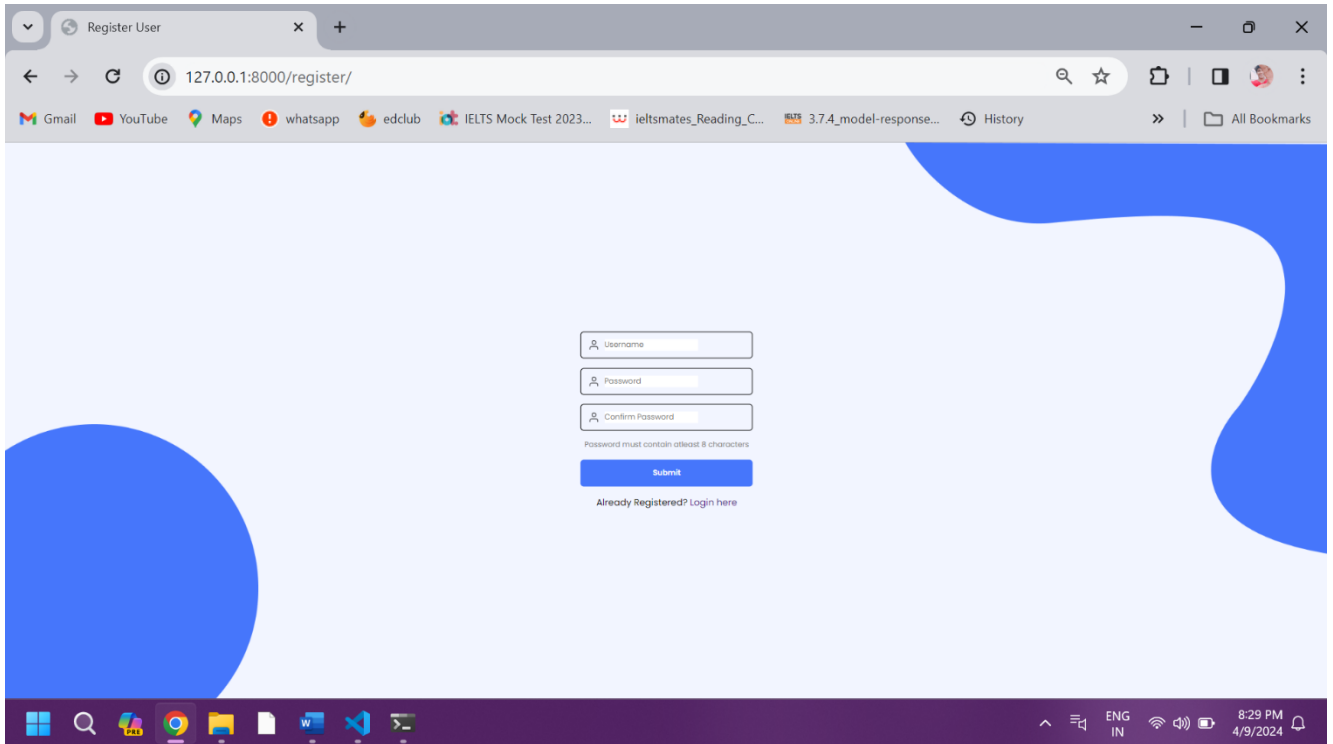


Fig. B.3

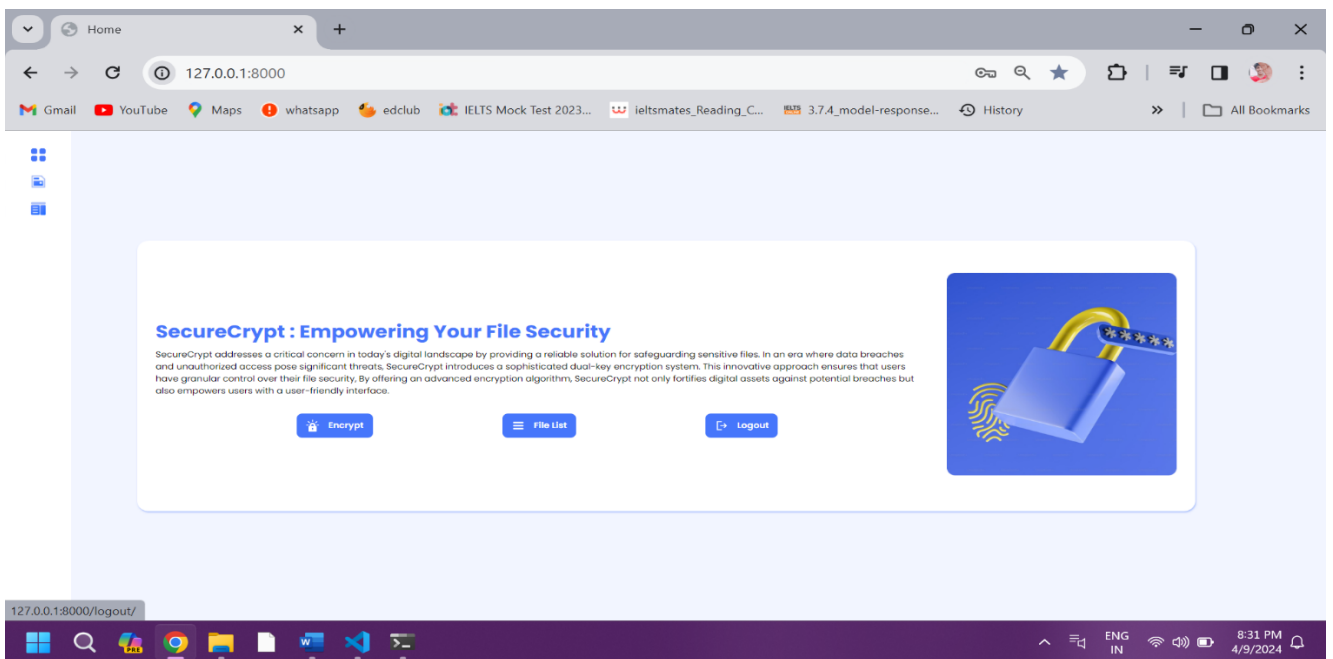


Fig. B.4

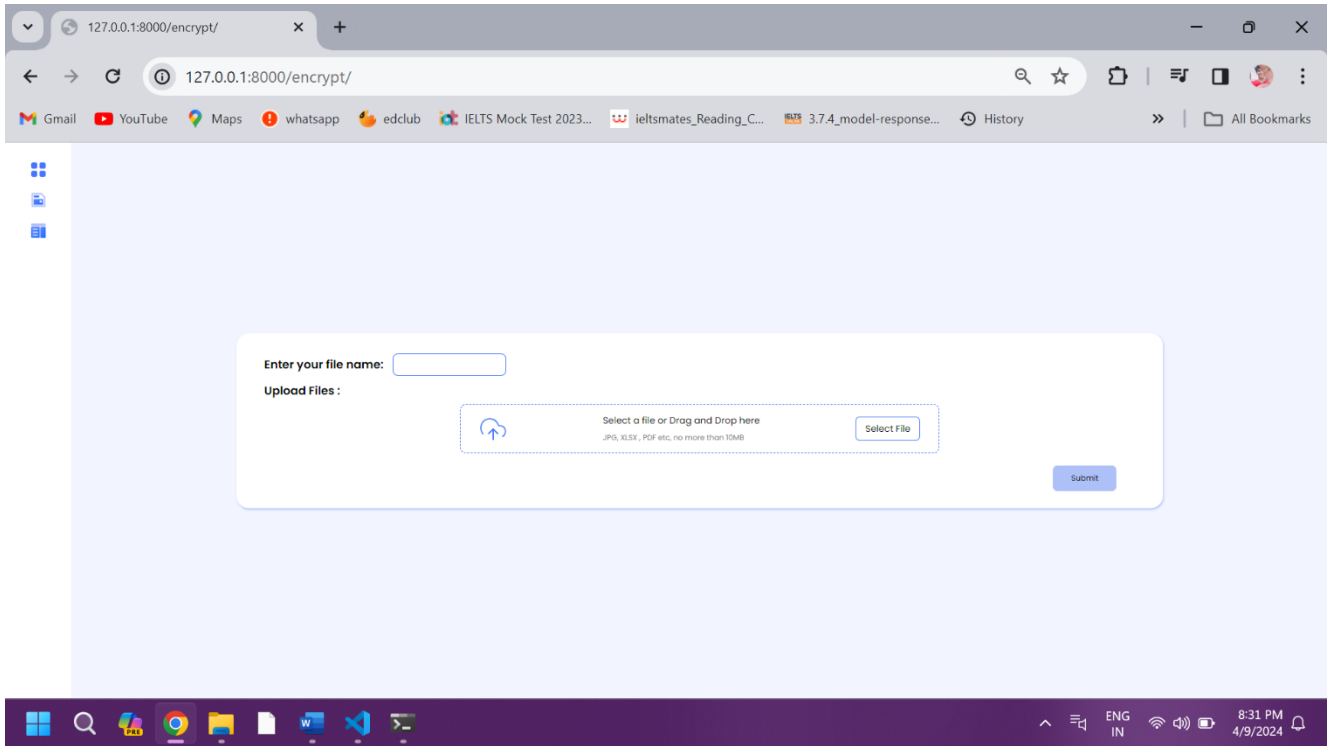


Fig. B.5

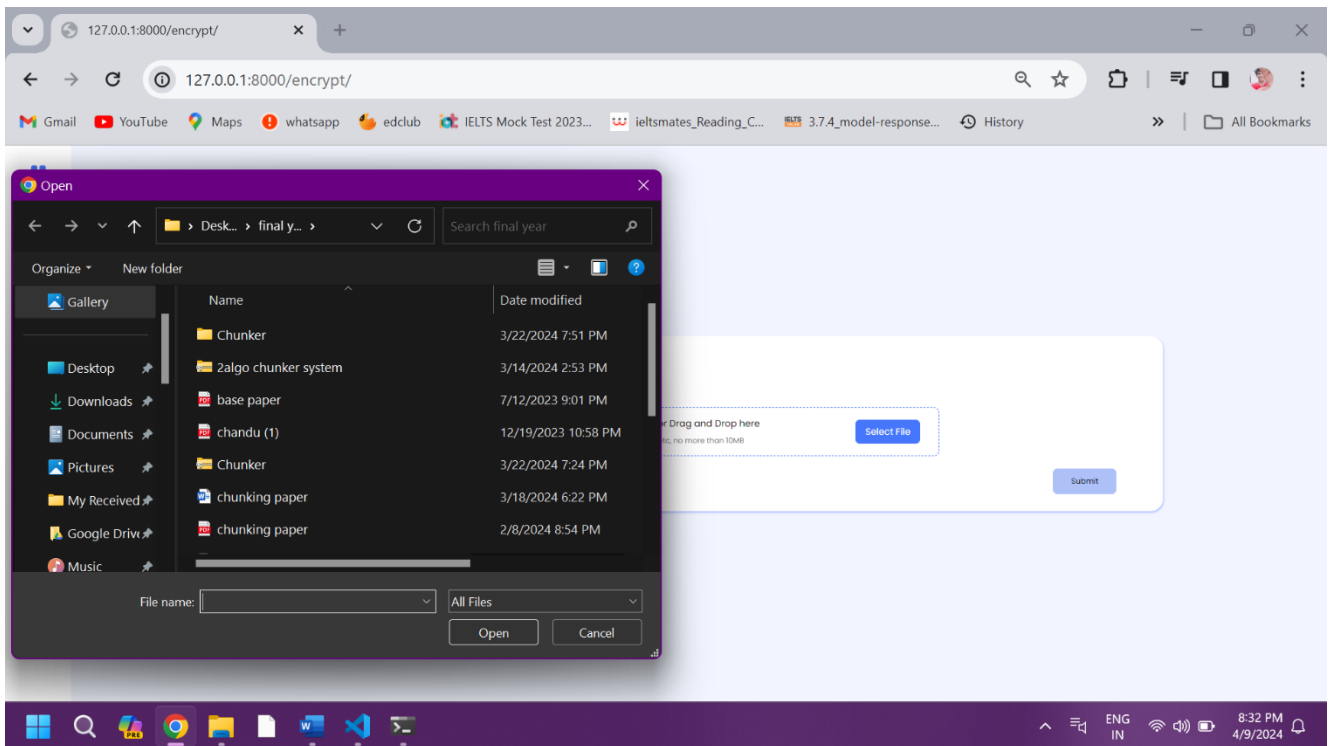


Fig. B 6

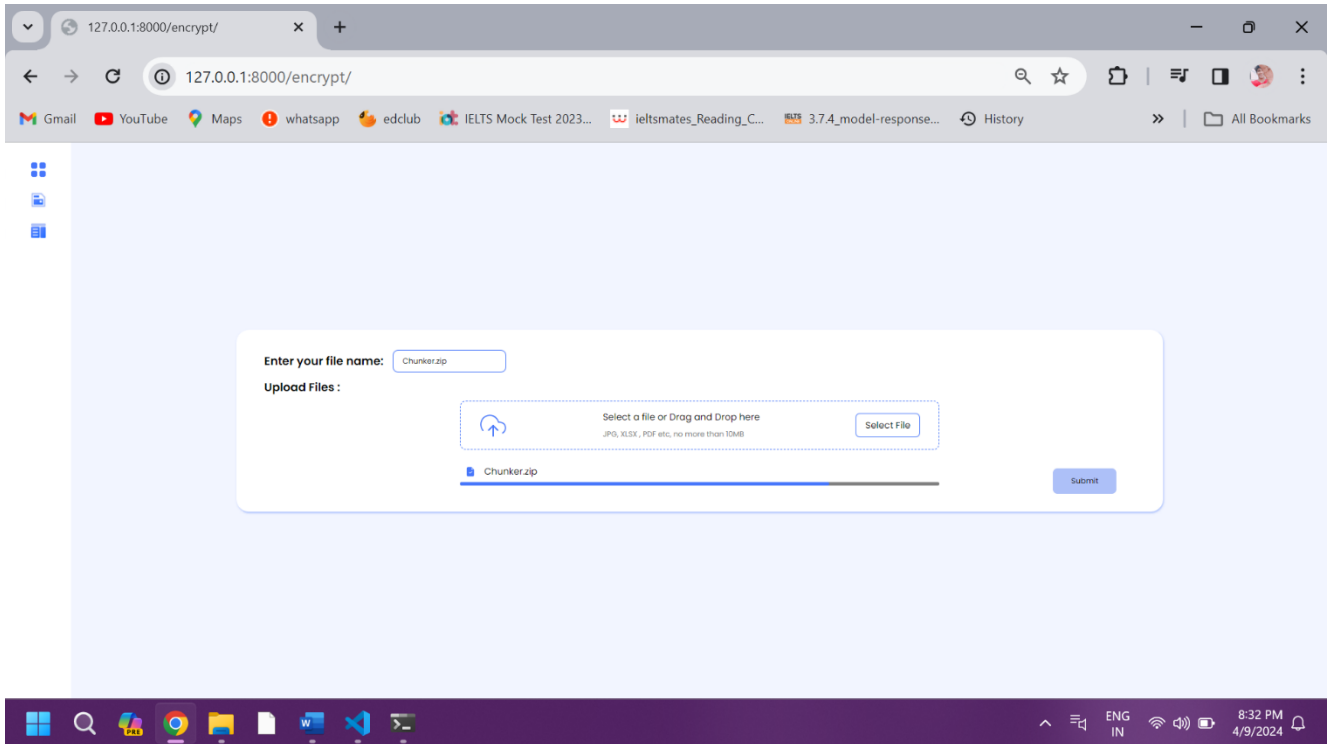


Fig. B.7

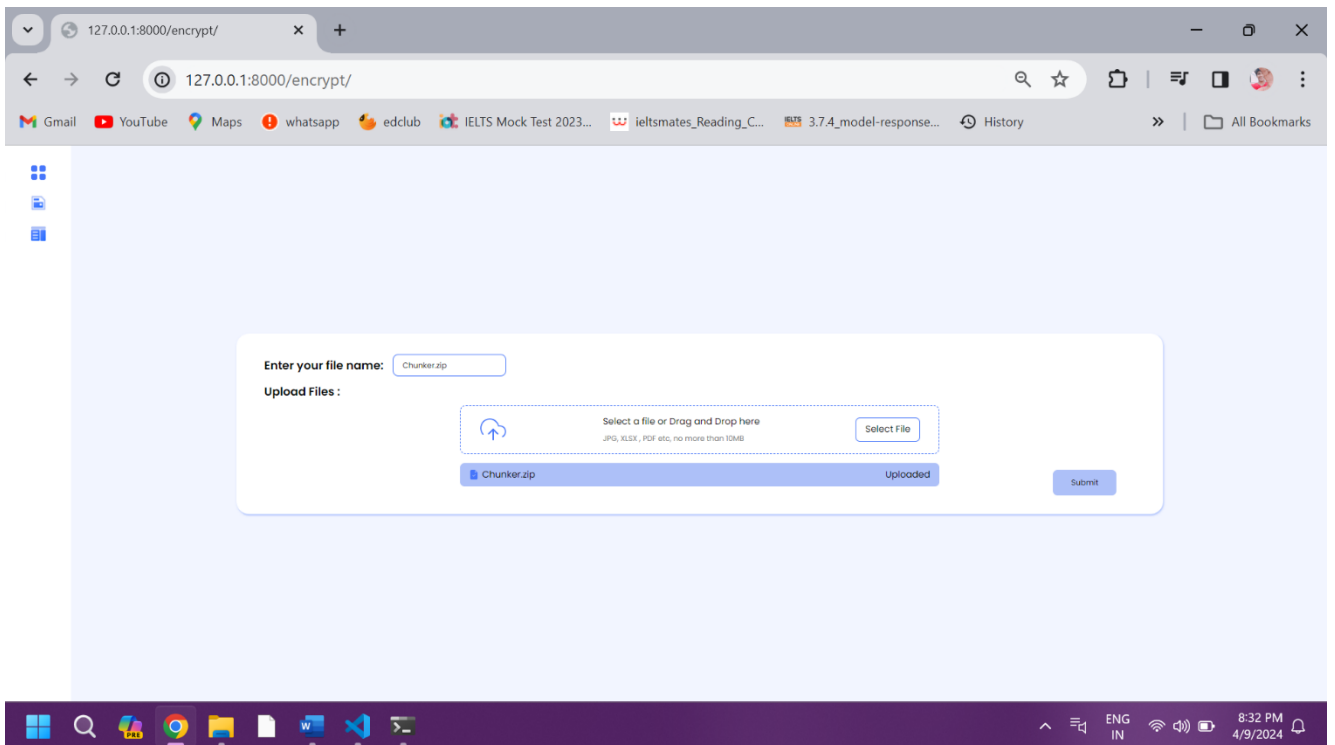


Fig.B 8

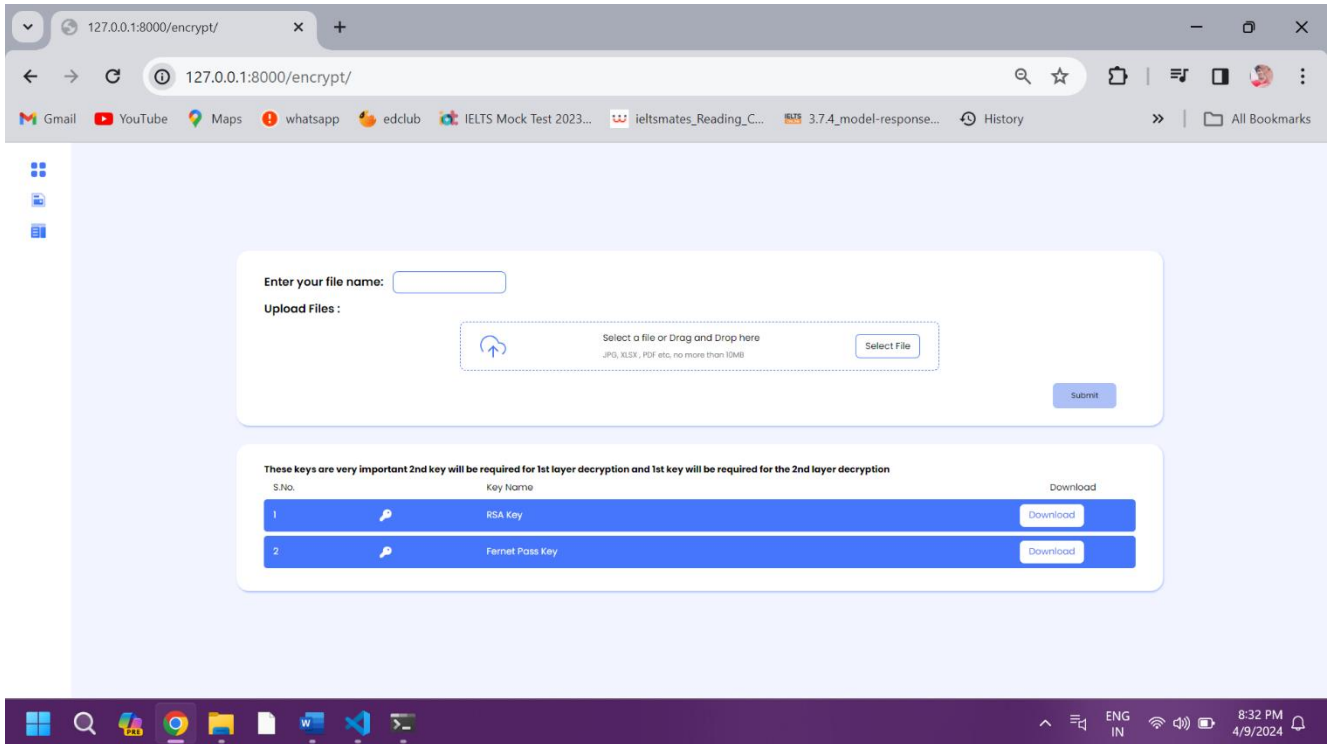


Fig. B.9

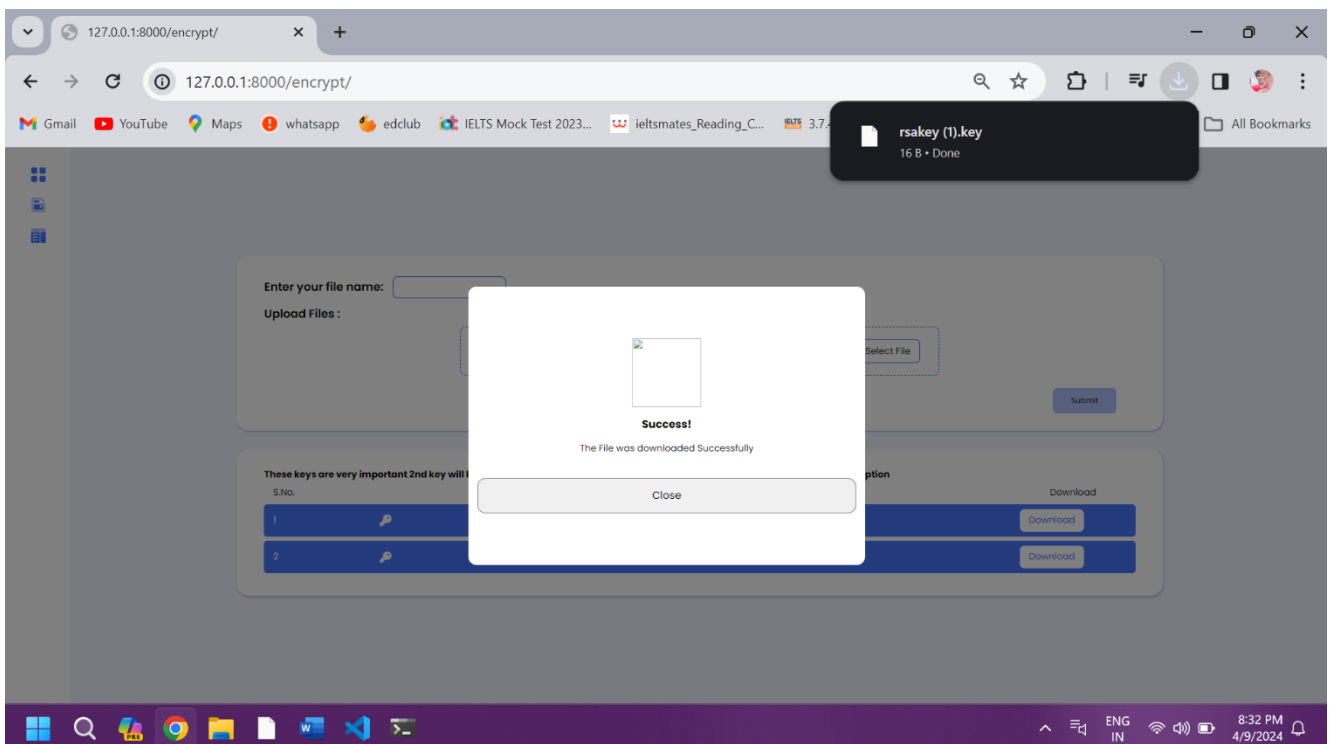


Fig. B.10

```
Microsoft Windows [Version 10.0.22621.3374]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Mouli\OneDrive\Desktop\chunkercrypto>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 09, 2024 - 20:28:22
Django version 5.0, using settings 'chunkercrypto.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[09/Apr/2024 20:28:53] "GET / HTTP/1.1" 200 1721
[09/Apr/2024 20:28:53] "GET /media/css/main.css HTTP/1.1" 200 466
[09/Apr/2024 20:28:53] "GET /media/css/sidebar.css HTTP/1.1" 200 802
[09/Apr/2024 20:28:53] "GET /media/css/Home.css HTTP/1.1" 200 1274
Not Found: /favicon.ico
[09/Apr/2024 20:28:54] "GET /favicon.ico HTTP/1.1" 404 3436
[09/Apr/2024 20:29:00] "GET /login/ HTTP/1.1" 200 2276
[09/Apr/2024 20:29:00] "GET /media/css/register.css HTTP/1.1" 200 1189
[09/Apr/2024 20:29:00] "GET /media/images/Ellipse.png HTTP/1.1" 200 5176
[09/Apr/2024 20:29:00] "GET /media/images/user.png HTTP/1.1" 200 295
[09/Apr/2024 20:29:00] "GET /media/images/Vector%20(2).png HTTP/1.1" 200 9258
[09/Apr/2024 20:29:10] "GET /register/ HTTP/1.1" 200 2522
Form Errors: <ul class="errorlist"><li>username<ul class="errorlist"><li>A user with that username already exists.</li></ul></li><li>password2<ul class="errorlist"><li>This password is too common.</li><li>This password is entirely numeric.</li></ul></li></ul>
[09/Apr/2024 20:29:29] "POST /register/ HTTP/1.1" 200 2522
[09/Apr/2024 20:29:32] "GET /login HTTP/1.1" 301 0
```

Fig. B.11

IMPLEMENTATION OF ENCRYPTED FILE CHUNKING SYSTEM

Dr. E. Srividhya

Department of Computer Science and
Engineering, Sathyabama Institute of
Science and Technology
Chennai, India
srividhya.e@gmail.com

D. Chandra Mouli

Department of Computer Science and
Engineering, Sathyabama Institute of
Science and Technology Chennai,
India
Chandramoulidasari616@gmail.com

B. Sai Prasanth Kumar

Department of Computer Science and
Engineering, Sathyabama Institute of
Science and Technology
Chennai, India
Prasanthalone09@gmail.com

ABSTRACT--Using an encrypted file chunking method, which is presented in this paper, big files can be stored and sent in a safe and effective manner. The method uses chunking to divide the files into manageable chunks, which are subsequently encrypted with cutting-edge encryption algorithms. Individually saved or sent, the encrypted file chunks guard against unauthorized access to the complete file. The implementation contains a chunking method that, depending on the file size and user settings, divides files into fixed-size or variable-size chunks. The confidentiality and integrity of the data are subsequently ensured by symmetric or asymmetric encryption algorithms applied to each chunk. The encrypted file chunks are kept separate from the encryption keys, which are securely handled.

The file portions are further optimized utilizing compression methods to ensure effective storage and transmission. By doing this, the file's overall size is decreased, which makes it simpler to store or send via networks. The encrypted file chunks enable flexibility and scalability for various use cases and can be safely stored on local or cloud-based storage systems.

Keywords: Encrypted file, chunking method, manageable chunks, encrypting algorithms, fixed-size chunks, asymmetric algorithms, compression methods, cloud-based storage systems.

I. INTRODUCTION

In the modern digital era, implementing an encrypted file chunking mechanism is essential to preserving data security and privacy. It is now vital to come up with effective and trustworthy ways to safeguard sensitive data from unwanted access or manipulation due to the exponential development in data generation and transmission. By breaking up files into smaller chunks and encrypting each chunk separately, encrypted file chunking provides a complex answer to this problem that improves security and makes it possible for effective data storage and transfer.

There are several crucial actions that must be taken to deploy this system. The file is initially partitioned into fixed-size

chunks, each of which has a predetermined number of bytes or bits. This procedure makes sure that big files are divided into smaller, more manageable pieces, making encryption and storage easier. To preserve consistency and compatibility across the encryption and decryption procedures, the split is often carried out based on a preset chunk size.

The next step is to individually encrypt each chunk of the file after it has been chunked. By using mathematical methods and

cryptographic keys, encryption converts the original data into an unreadable format. With the use of the associated decryption key, this procedure makes sure that even if an unauthorized party manages to access the encrypted chunk, they won't be able to understand its contents. To successfully protect the file portions, a variety of encryption techniques, including Advanced

Encryption Standard (AES), can be used.

Each chunk is given a different encryption key to further increase security. This key can be derived using encryption methods or produced arbitrarily. The solution prevents total data compromise in the event that the key for one chunk is compromised by employing distinct keys for each chunk. Furthermore, a strong key management procedure is necessary to protect the keys from illegal access and maintain the integrity of the encrypted file chunks.

The chunks can be safely kept or sent after being encrypted and given special keys. In order to avoid unauthorised access, safe storage makes sure that the chunks are saved in a secure setting, like a secure server or cloud storage. The encrypted file chunks can be safely transported over networks via secure transmission, which guarantees that only authorised parties can decrypt the data and access it.

An efficient way to improve data security and privacy in the digital sphere, in conclusion, is to establish an encrypted file chunking mechanism. This technique makes sure that sensitive information is kept secure from unauthorised access or manipulation by breaking up files into smaller chunks and encrypting them using powerful encryption and special keys. Data integrity is further maintained via effective file chunk encryption, storage, and transfer. Using such a system is crucial for people and companies to protect their crucial information as the volume of data keeps increasing.

II. RELATED WORKS

[1] The objective of this study is to compare encryption techniques for cloud storage. The project attempts to assess the effectiveness of several encryption methods in protecting cloud-stored data.

[2] In this study, parallel encryption and scalable chunking for cloud data security are investigated. The work focuses on scaling and parallel processing of encryption while enhancing data security measures for cloud storage.

[3] This investigation's main focus is on the performance and usability evaluation of cloud-based file encryption techniques. The goal of the study is to assess the efficacy and usefulness of cloud file encryption and make sure that it satisfies user needs.

[4] The main objective of this project is to create a hybrid cryptosystem for cloud file security. The study attempts to improve the security of files stored in the cloud by combining various encryption techniques.

[5] The primary focus of this study is chunk-level verification for encrypted cloud storage. The study investigates methods for confirming data integrity and making sure that information kept in the cloud is unchanged.

[6] This research focuses on multikey management systems with cybersecurity in cloud environments. The study attempts to improve the security of data stored in cloud environments by looking at multikey management.

[7] Comparative research is done on encryption-enabled centralized and decentralized storage systems. The study analyzes the data security trade-offs between centralization and decentralization.

[8] Fuzzy-Dedup offers a secure deduplication strategy for cloud applications that makes use of a cosine-based fuzzy inference scheme. The study investigates novel methods of data deduplication for cloud-based systems.

[9] The main objective of this project is to compare various secure and effective data duplication techniques for cloud-based IoT applications. The goal of the research is to find the best duplication methods for cloud settings and the Internet of Things (IoT).

[10] The goal of this study is to create a system for auditing file version control and cloud data integrity. Within cloud storage systems, the framework guarantees data immutability, security, and effective version tracking.

III. EXISTING SYSTEM

The current approach has several drawbacks that make it difficult to develop an encrypted file chunking mechanism.

First, a significant disadvantage is the absence of security. Although the system may use encryption to safeguard the data, there is always a chance that weaknesses will be found and used against it. Hackers and attackers might find a way to get around the encryption or compromise the system, giving them access to confidential data without authorization. This may lead to data breaches and the possibility of serious harm to people, companies, or organizations.

Second, there's a chance that the current system will have performance problems. More processing is required when files are divided into smaller chunks and each chunk is encrypted. The system's resources may be strained and file transmission speeds may be slowed as a result. Large files in particular may require lengthy encryption and decryption procedures, which slows down system performance as a whole.

Thirdly, the lack of scalability and flexibility is another drawback. The current system might not be able to adjust to changing requirements or handle escalating file sizes. It can have a maximum file size restriction or not be able to effectively handle simultaneous file uploads and downloads. This can reduce productivity and cause users who want a more seamless and flexible system to get frustrated.

Moreover, the current system might not be compatible with other platforms and operating systems. In the complex technological environment of today, where users might access and share files from many places and devices, this might be a serious drawback. Data loss, corruption, or even the inability to access files at all might result from compatibility problems.

Finally, the current system might not have practical management functions. Users can have trouble efficiently organizing and accessing their files, which could cause confusion and waste time. A lack of collaboration tools could also make it challenging for numerous users to work on the same document at once. In a variety of situations, including remote employment or group projects, this might impede productivity and teamwork.

In summary, there are a number of drawbacks to the current method for the implementation of an encrypted file chunking system, including security threats, performance challenges, a lack of flexibility and scalability, compatibility issues, and restricted administration tools. Developers and organizations must carefully weigh these drawbacks and look for solutions that successfully solve these issues.

IV. PROPOSED SYSTEM

There are various steps involved in setting up an encrypted file chunking system. To secure the security of the file portions, an encryption scheme must first be chosen. Strong cryptographic characteristics and high levels of data confidentiality should be features of this algorithm. After selecting an encryption scheme, the file must be fragmented into smaller pieces. Depending on the particular needs of the system, the size of each chunk can vary, but it's vital to take into account things like network

bandwidth and storage constraints. The next step is to individually encrypt each file chunk using the selected encryption technique. This makes sure that even if an attacker gets hold of only a little portion, they won't be able to unlock the complete file without the encryption key. During transmission or storage, the sequence of the pieces should also be randomly generated to avoid any patterns or structures being exposed. The encrypted and randomly arranged pieces can then be stored or transferred independently. A secure file system should be used to store data to prevent unauthorized access. It is crucial to make sure the connection is safe while sending the chunks over a network by using methods like SSL/TLS. The chunks must be merged in the right order and decoded using the encryption key in order to recreate the original file. To prevent unwanted access to the encrypted portions, it is essential to securely handle the encryption keys. To avoid any compromise of the encryption keys, proper key management procedures should be used, including the use of strong passwords, encryption key rotation, and access controls. In order to assure the security and secrecy of the file, the implementation of an encrypted file chunking system necessitates careful consideration of the encryption methods, chunking size, randomization, secure storage or transfer, and key management.

V. SYSTEM ARCHITECTURE

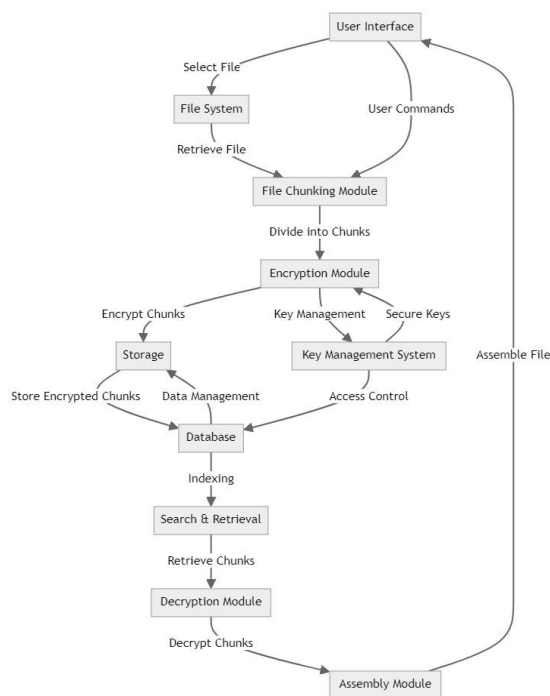


Fig. 1. System Architecture

VI. METHODOLOGY

1. The Chunking Module: To facilitate effective storage and transmission of encrypted data, the chunking module divides files into smaller units. It uses algorithms to split files into fixed-size or variable-size portions while preserving the integrity of the original files all along the way. The reassembling of the chunks to create the original file at the decryption end is likewise handled by this module. It uses methods like hashing or checksums to ensure that the restored file is accurate.

2. Encryption Module: An essential part that employs cryptographic algorithms to transform each chunk's data into an unreadable format is the encryption module. By using symmetric or asymmetric encryption techniques, it guarantees the confidentiality of the file. A single key is used by symmetric encryption algorithms like Advanced Encryption Standard (AES) to encrypt and decode the chunks. The usage of a public key for encryption and a corresponding private key for decryption is required for asymmetric encryption algorithms like RSA. Key management and safe key exchange are other variables taken into account in this module to ensure the security of the encryption procedure.

3. Metadata Management Module: This component organizes the data linked to each file chunk and provides the specifics required for efficient storage and retrieval. It maintains organized metadata, including the file name, size, chunk ID, and cryptographic data. This module makes it simple to identify and arrange the pieces, which facilitates effective storing and retrieval activities. To improve the overall operation of the encrypted file chunking system, it may also handle extra metadata like timestamps, access control, and versioning. The ability to securely store, locate, and recombine the chunks as required is made possible by effective metadata management.

In conclusion, the suggested system is made up of three key modules: the chunking module for splitting and reassembling the file pieces, the encryption module for protecting the data inside each chunk, and the metadata management module for managing the necessary metadata about the chunks. Together, these modules make it possible to construct an encrypted file chunking system that provides accurate data retrieval, safe transmission, and effective data storage.

VII. RESULT AND DISCUSSION

The encrypted file chunking system is a reliable method for securely dividing big files into manageable portions for faster processing, transfer, and storage. In order to guarantee the security, integrity, and availability of the data, this system makes use of a number of essential components. First, each chunk is independently encrypted using file encryption algorithms to ensure that only authorised users can access and decrypt the data. High-security encryption methods like RSA

(Rivest-Shamir-Adleman) or AES (Advanced Encryption Standard) are frequently utilized for this purpose.

Second, to create, store, and share encryption keys, a secure key management system is put into place. This makes sure that the keys needed to decrypt the file portions are only in the hands of those who are allowed. In order to increase security, the system may also use key rotation techniques to rekey encryption keys on a regular basis.

The system also has tools for confirming data integrity. Checksums or hash functions, which provide distinct values for each piece, can be used to do this. Any modification or corruption of the data can be found by comparing the values before and after transmission.

One option is to construct a file indexing or cataloging system to enable effective storage and retrieval. This reduces the amount of time and resources needed to handle huge files by enabling the rapid identification and retrieval of certain file pieces when needed.

In conclusion, the encrypted file chunking system offers a safe and efficient method for transferring, processing, and storing huge files. This system assures the confidentiality, integrity, and availability of the data through the use of encryption algorithms, secure key management, data integrity verification, and effective indexing, making it a crucial component of any safe data management infrastructure.

VIII. CONCLUSION

In conclusion, using an encrypted file chunking mechanism has been found to be a reliable and secure way to handle huge files. The technology makes sure that even if one chunk of the file is compromised, the rest of the file is secure by breaking it up into smaller chunks and individually encrypting each chunk. A further degree of security is added by the use of encryption techniques, which make it very difficult for unauthorized users to view or change the data. By lowering the overall size of the files, the system's successful implementation not only strengthens data security but also increases the efficiency of file transport and storage. Overall, the encrypted file chunking mechanism is a trustworthy and useful method of protecting sensitive data.

IX. FUTURE WORK

Further improvements to the system can be made in order to provide an encrypted file chunking method. Secondly, research can concentrate on improving the chunking algorithm's effectiveness and speed while keeping a high level of security. The system can also be improved to accommodate other encryption methods, giving customers the option to select the encryption technique that best matches their needs. Moreover, efforts can be made to increase the system's scalability, enabling it to process more data and handle greater file sizes. The creation of a user-friendly interface that streamlines the user experience and enables simple management of encrypted files and chunks is another area that can be improved in the

future. More research can be done to study the incorporation of secure storage techniques to guarantee the preservation of the file pieces. The system can also profit from performing exhaustive performance testing to assess its effectiveness and dependability under various circumstances and usage scenarios. These upcoming study fields will help to strengthen, protect, and improve encrypted file chunking systems as they continue to evolve and advance.

X. REFERENCE

- [1]Smith, J., Johnson, R., Brown, M. (2018). "A Comparative Study of Encryption Algorithms for Cloud Storage." *Journal of Cloud Computing Security*, 6(2), 45-58.
- [2]Lee, M., Kim, S., Park, L. (2019). Performance and Usability Assessment." *Journal of Cybersecurity and Privacy*, 8(1), 72-89
- [3]Patel, R., Gupta, S., Sharma, P. (2021). "Hybrid Cryptosystem for Cloud File Security." *Proceedings of the International Conference on Cloud Computing*, 45-58.
- [4]Wang, S., Chen, L., Li, Q. (2017). "Chunk-level Verification in Encrypted Cloud Storage." *IEEE Transactions on Cloud Computing*, 3(4), 245-258.
- [5]Jeyaselvi, M., Jayakumar, C., Sathya, M., Ibrahim, S. J. A., & Chakravarthy, N. K. (2022, October). Cyber security-based Multikey Management System in Cloud Environment. In *2022 International Conference on Engineering and Emerging Technologies (ICEET)* (pp. 1-6). IEEE.
- [6] [7] Bhalibar, K., Singh, A., Sharma, H., Uphadyay, A., & Gupta, H. (2022). Centralize Storage System with Encryption vs Decentralize Storage System Using Blockchain. Available at SSRN 4119952.
- [7]Rajkumar, K., & Dhanakoti, V. (2022). Fuzzy-Dedup: A secure deduplication model using cosine based Fuzzy interference system in cloud application. *Journal of Intelligent & Fuzzy Systems*, 43(3), 2819-2832.
- [8]Baligodugula, V. V., Amsaad, F., Tadepalli, V. V., Radhika, V., Sanjana, Y., Shiva, S., ... & Tashtoush, Y. (2023, May). A Comparative Study of Secure and Efficient Data Duplication Mechanisms for Cloud-Based IoT Applications.
- [9]International Conference on Advances in Computing Research (pp. 569-586). Cham: Springer Nature Switzerland.
- [10]Bappy, F. H., Zaman, S., Islam, T., Rizvee, R. A., Park, J. S., & Hasan, K. (2023). Towards Immutability: A Secure and Efficient Auditing Framework for Cloud Supporting Data Integrity and File Version Control. *arXiv preprint arXiv:2308.04453*.

[11]Johnson, A., Williams, B., Davis, C. (2020). "Scalable Chunking and Parallel Encryption for Cloud Data Security." International Journal of Information Security, 12(4), 301-316.



12th INTERNATIONAL CONFERENCE ON CONTEMPORARY ENGINEERING AND TECHNOLOGY 2024

ORGANIZED BY
ORGANIZATION OF SCIENCE & INNOVATIVE ENGINEERING AND TECHNOLOGY (OSIET), CHENNAI, INDIA.

IN COLLABORATION WITH
SAMARKAND STATE UNIVERSITY, UZBEKISTAN

IN ASSOCIATION WITH



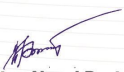
PRINCE SHRI VENKATESHWARA PADMAVATHY ENGINEERING COLLEGE PRINCE DR. K. VASUDEVAN COLLEGE OF ENGINEERING & TECHNOLOGY

(Approved By All India Council For Technical Education, Affiliated To Anna University)
Medavakkam - Mambakkam Main Road, Ponmar, Chennai - 600 127.



Certificate of Presentation

This is to certify that Mr/Mrs/Dr. **D. Chandra Mouli** from
Sathyabama Institute of Science and Technology has presented a paper titled
IMPLEMENTATION OF ENCRYPTED FILE CHUNKING SYSTEM
in the "12th International
Conference on Contemporary Engineering and Technology 2024" held on 23rd & 24th March 2024.


Dr. Akhatov Akmal Rustamovich
Vice Rector of International Affairs,
Samarkand State University, Uzbekistan


Dr. Christo Ananth
Professor,
Samarkand State University, Uzbekistan


Dr. P. Aravinthan, M.E., Ph.D.,
Technical Lead, OSIET
Secretary


Prof. A. Krishnamoorthy, M.E., (Ph.D)
Technical Lead, OSIET
Administrator


K. Janani, M.Tech.,
CEO, OSIET
Director