

# **Data Storage and Sharing Solution with Guaranteed Reliability**

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

By

**P MADHAN VENKATA REDDY (Reg .No - 39110809 )**

**MVR VIKAS (Reg. No – 39110592 )**



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

# **SATHYABAMA**

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

**Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by**

**AICTE JEPPIAAR NAGAR, RAJIV GANDHI SALAI,**

**CHENNAI - 600119**

**APRIL - 2023**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

[www.sathyabama.ac.in](http://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 **P MADHAN VENKATA REDDY(Reg. No- 39110809)** and **MVR VIKAS(Reg. No - 39110592)** and who carried out the Project Phase-2 entitled **"Data Storage and Sharing Solution with Guaranteed Reliability"** under my supervision from January 2023 to April 2023.

Internal Guide  
Ms.R.ASHA M.E.,(Ph.D)

Head of the Department  
Dr. L. LAKSHMANAN, M.E., Ph.D



Submitted for Viva voce Examination held on 20.04.2023

Internal Examiner

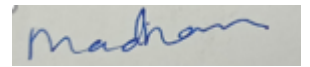
External Examiner

## DECLARATION

I, **P MADHAN VENKATA REDDY (Reg. No- 39110809)**, hereby declare that the Project Phase-2 Report “**Storage and Sharing Solution with Guaranteed Reliability**” entitled done by me under the guidance of **Ms. R. ASHA, 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:20.04.2023**

**PLACE: Chennai**



**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

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

I convey my 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 to my Project Guide **Ms. R. ASHA M.E., (Ph.D.)**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 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**

An enterprise that requires essential digital data might select from storage and sharing options that have been approved by an established Internet provider. There are several issues to consider: The organization's identity be protected on certificates and sensitive digital data stored on the system be protected can data sharing be made secure, open, and equitable In what ways can the validity of shared data be verified without compromising privacy. To this end, we offer a variety of formats for creating, storing, and sharing data. We implement a group signature system for a collection of trustworthy businesses that offer the same kind of service in the data- producing schema, and each business in the collection produces valuable digital data from the raw data received from the others. Data owners upload their data to the Inter- Planetary File System network and store the corresponding access address and certificate on the blockchain ledger using the data storage schema. In accordance with the data sharing schema, everyone could verify the reliability of shared data before requesting data sharing with the data owner Smart contracts are used to share data, and to ensure honesty, so that the involved parties should escrow. Among the security properties guaranteed by schemas for storing and sharing data are confidentiality, anonymity, integrity, and privacy.

### **Key Words:**

Distributed Ledger Technology (DLT)

Meaningful Data (MD), Reputable Organization (RO)

Dataowner(DO)

DecentralizedStorage.

<b>TABLE OF CONTENTS</b>		
<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	v
	<b>LIST OF FIGURES</b>	vii
1	<b>INTRODUCTION</b>	1
2	<b>LITERATURE SURVEY</b>	4
	2.1 Inferences from literature survey	4
3	<b>REQUIREMENTS ANALYSIS &amp; FEASIBILITY STUDY</b>	7
	3.1 Existing Method	7
4	<b>PROPOSED SYSTEM &amp; ARCHITECTURE</b>	8
	4.1 Proposed System	8
5	<b>IMPLEMENTATION DETAILS</b>	9
	5.1 Data Encryption	10
	5.2 SHA Algorithm	11
	5.3 System Specifications	17
	5.4 UML Diagram	20
6	<b>RESULTS &amp; DISCUSSION</b>	31
7	<b>CONCLUSION</b>	32
	<b>REFERENCES</b>	33
	<b>APPENDIX</b>	35

A. Source Code	35
B. Screenshots	55
C. Research Paper	66

<b>LIST OF FIGURES</b>		
<b>FIGURE NO.</b>	<b>FIGURE NAME</b>	<b>PAGE NO.</b>
4.1	Raw Idea Of Architecture	8
5.1	Sender to Receiver	9
5.2	Hash Function	12
5.3	Waterfall Model	13
5.4	UseCase Diagram	22
5.5	Class Diagram	23
5.6	Sequence Diagram	23
5.7	Collaboration Diagram	24
5.8	Deployment Diagram	25
5.9	Activity Diagram	26
5.10	Component Diagram	27
5.11	ER Diagram	28
5.12	DFD Diagram	29
5.13	DFD Diagram Level 1	30



# CHAPTER 1

## INTRODUCTION

There has been exponential data growth in the world, and trusted data are considered one of the most valuable assets of individuals and organizations. The amount of data created and stored globally are predicted to create about 175 zettabytes by 2025. It is also estimated that by 2025 the global consumers interacting with data everyday will reach 5 billion. Consequently, the demand for valuable data storing and sharing is tremendous, which also poses challenges related to data security in the processes of data storing and sharing. Currently, there are two main architectures used for data storing and sharing, centralized and decentralized architectures

For the centralized architecture, organizations can store data on their datacentre system. However, these systems have high operating costs and are limited in scalability. Using cloud storage services can reduce costs and can be flexible in system expansion, and more suitable for IoT systems. The combination of IoT and cloud storage services is a matter of studies. To protect the security and privacy of data storing and sharing, encryption algorithms and access control models are proposed in Murat Kantarcioglu et al. proposed SECURED L for protecting the sensitive data stored in databases. However, the centralized architecture has two limitations including data security, stored data could be accessed, modified, or removed illegally by system administrators or attackers who compromised the system; availability, when the centralized systems are crashed due to system overload, denial-of-service or distributed denial-of-service (DoS/DDoS) attacks, or system errors, the services are not available for users.

For the decentralized architecture, most solutions use blockchain (BC) technology as the main component in the systems because of its properties such as anonymity, transparency, decentralization, and auditability. However, current solutions do not provide features for verifying the accuracy and the reliability of the shared data on the BC network. Specifically, data verified and

certified by a reputable organization (RO) are considered as meaningful data (MD). For instance, in the medical field, a diagnostic result of an electronic medical record is published by a reputable medical organization with highly skilled doctors, which is MD. In the education field, a lecture that is assessed and certified by a professional board of a reputable university is MD. MD needs to be securely stored on the system, besides a data owner (DO) can completely share or commercialize his/her MD to other people or organizations on the network. Data sharing methods must ensure that requesters can verify the reliability and accuracy of shared data before deciding to perform a data-sharing contract. The accuracy and the reliability of the shared data on the BC network. Specifically, data verified and certified by a reputable organization (RO) are considered as meaningful data (MD). For instance, in the medical field, a diagnostic result of an electronic medical record is published by a reputable medical organization with highly skilled doctors, which is MD. In the education field, a lecture that is assessed and certified by a professional board of a reputable university is MD. MD needs to be securely stored on the system, besides a data owner (DO) can completely share or commercialize his/her MD to other people or organizations on the network. Data sharing methods must ensure that requesters can verify the reliability and accuracy of shared data before deciding to perform a data-sharing contract

With the traditional data sharing method, the integrity of shared data is based on trust between the two partners participating in the exchange process. For example, doctors/hospitals absolutely believe that medical records received from their patients are integrity. In some cases, RO needs to ensure anonymity in MD generated by themselves. And the privacy of DO also needs to be protected as they don't want anyone to know which RO's service they used. In addition, the identities of those involved in the sharing process also need to be anonymous; and shared data need to be verified the reliability while still ensuring the privacy of its content.

Data storing and sharing for certified digital data are very necessary, which requires data storage and sharing solutions that need to meet all of the following requirements:

**Data storing:** The anonymity of certificate authorities and the privacy of DO on stored data must be protected.

**Data sharing:** Everyone on the system can verify the reliability of shared data before submitting a sharing request to DO. Note that everyone can only verify the reliability of the shared data but cannot read its contents. The data sharing process is done directly between and DU without depending on any intermediaries. The system serving data storage and sharing must ensure availability, integrity, and scalability. However, current solutions do not meet all of the above requirements. In this paper, we propose data producing, data storing, and data sharing schemes. We consider RO as a data provider (DP), and DPs providing the same type of service join in a group. In the data producing scheme, a group manager sets up a group of DPs that provide the same type of service. A raw data of DO is produced into MD by a particular DP in the group. Then, DP encrypts MD using a symmetric algorithm along with a secret key. Later, DP generates a certificate on the MD ciphertext (denoted by EMD). Finally, EMD, the certificate, and DP's information will be sent to DO through a secure channel. In the data storing scheme, DO stores EMD on Inter-Planetary File System (IPFS), the access address of EMD on IPFS and related information are stored in a transaction on the blockchain system. However, current solutions do not meet all of the above requirements. In this paper, we propose data producing, data storing, and data sharing schemes. We consider RO as a data provider (DP), and DPs providing the same type of service join in a group. In the data producing scheme, a group manager sets up a group of DPs that provide the same type of service. A raw data of DO is produced into MD by a particular DP in the group. Then, DP encrypts MD using a symmetric algorithm along with a secret key. Later, DP generates a certificate on the MD ciphertext (denoted by EMD). Finally, EMD, the certificate, and DP's information will be sent to DO through a secure channel. In the data storing scheme, DO stores EMD on Inter-Planetary File System (IPFS), the access address of EMD on IPFS and related information are stored in a transaction on the blockchain system.

# LITERATURE SURVEY

## 2.1 CHAPTER 2

## INFERENCES FROM LITREATURE SURVEY

**M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, “Data center network virtualization: A survey,” *EEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart., 2013** It present a survey of the current state of-the-art in data center networks virtualization, and provide a detailed comparison of the surveyed proposals. We discuss the key research challenges for future research and point out some potential directions for tackling the problems related to data center design.

**L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, “An IoT-oriented data storage framework in cloud computing platform,” *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1443–1451, May 2014**

A data storage framework not only enabling efficient storing of massive IoT data, but also integrating both structured and unstructured data. This data storage framework is able to combine and extend multiple databases and Hadoop to store and manage diverse types of data collected by sensors and RFID readers. In addition, some components are developed to extend the Hadoop to realize a distributed file repository, which is able to process massive unstructured files efficiently.

**T. A. Phan, J. K. Nurminen, and M. Di Francesco, “Cloud databases for Internet-of-Things data,” in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom)*, Sep. 2014, pp. 117–124.**

The proposed solution includes strategies for expressing common IoT data in the form of key-value pairs, as well as a data pre-processing and sharing mechanism to build a Web of Things connecting HTTP-enabled smart devices such as sensors and actuators with virtual “things” such as services, social networks, and APIs. The platform adopts Mongo DB as database server, provides models and interfaces

that help to abstract and adopt different types of data and devices. This addressed the challenges related to data that are dynamic, various, massive, and spatial temporal (i.e., each sample corresponds to a specific time and location). To provide a uniform storage mechanism for heterogeneous sensor data, the system combined the use of the relational model and the key-value model, and was implemented with a PostgreSQL database. Its multi-layer architecture was claimed to reduce the amount of data to be processed at the cloud management layer. Besides, the work also provided several experiments that showed a promising performance when storing and querying a huge volume of data.

**K. Yasumoto, H. Yamaguchi, and H. Shigeno, “Survey of real-time processing technologies of IoT data streams,” J. Inf. Process., vol. 24, no. 2, pp. 195–202, 2016.**

The survey on emerging technologies toward the real-time utilization of IoT data streams in terms of networking, processing, and content curation and clarify the open issues. Then we propose a new framework for IoT data streams called the Information Flow of Things (IFoT) that processes, analyzes, and curates massive IoT streams in real-time based on distributed processing among IoT devices.

**A. Kumar, N. C. Narendra, and U. Bellur, “Uploading and replicating Internet of Things (IoT) data on distributed cloud storage,” in Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD), Jun. 2016, pp. 670–677.**

solution assumes the existence of multiple distributed cloud data centers, called mini-Clouds, among which data can be replicated model our problem comprehensively based on various parameters such as effective bandwidth of the IoT network, available number and size of data items at each mini-Cloud, and we present our problem as a collection of various sub-problems based on subsets of these parameters. We prove that the exact solution to the problem is intractable, and we present a number of heuristic strategies to solve it. Our results show that the performance of any heuristic is bounded by the read and write latency of mini Clouds and the best we can do is often 12 times the worst we can do for a given number of data items to be uploaded and replicated from the gateways to the mini Clouds in test setup.

**K. Hossain, M. Rahman, and S. Roy, “IoT data compression and optimization techniques in cloud storage: Current prospects and future directions,”*Int. J. Cloud Appl. Comput.*, vol. 9, no. 2, pp. 43–59, Apr. 2019.**

This article presents a detailed survey on different data compression and storage optimization techniques in the cloud, their implications, and discussion over future directions. The development of the smart city or smart home systems lies in the development of the Internet of Things (IoT). With the increasing number of IoT devices, the tremendous volume of data is being generated every single day. Therefore, it is necessary to optimize the system's performance by managing, compressing and mining IoT data for smart decision support systems. In this article, the authors surveyed recent approaches with up-to-date outcomes and findings related to the management, mining, compression, and optimization of IoT data. The authors then discuss the scopes and limitations of present works and finally, this article presents the future perspectives of IoT data management on basis of cloud, fog, and mobile edge computing

## **CHAPTER 3**

### **REQUIREMENTS ANALYSIS & FEASIBILITY**

#### **STUDY 3.1 Existing Method:**

Existing System provides the data sharing and data producing but there are no capabilities for validating the accuracy and reliability of shared data on the BC network in the solutions. Meaningful data is defined as data that has been validated and approved by a reputable organisation (RO) (MD). Data storing and sharing for certified digital data are very necessary, which requires data storage and sharing

#### **Disadvantages:**

- High complexity.
- Low Secure.
- Low Performance

The development of the smart city or smart home systems lies in the development of the Internet of Things (IoT). With the increasing number of IoT devices, the tremendous volume of data is being generated every single day. Therefore, it is necessary to optimize the system's performance by managing, compressing and mining IoT data for smart decision support systems. In this article, the authors surveyed recent approaches with up-to-date outcomes and findings related to the management, mining, compression, and optimization of IoT data. The authors then discuss the scopes and limitations of present works and finally, this article presents the future perspectives of IoT data management on basis of cloud, fog, and mobile edge computing

## CHAPTER 4

### PROPOSED SYSTEM & ARCHITECTURE

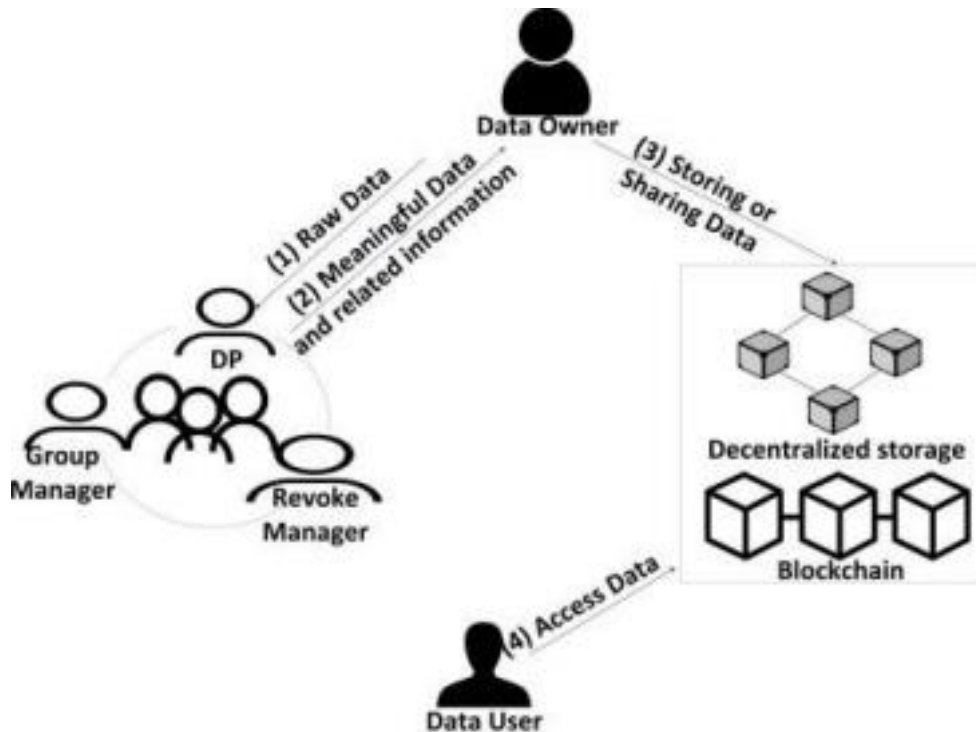


FIG 4.1 RAW IDEA OF ARCHITECTURE

#### 4.1 Proposed System:

Our proposed scheme for data production, data storage, and information sharing. We use a group authentication protocol in the data generating structure for a team of reputable companies that provide the identical sort of service. One of the organisations in the group produces valuable online information from raw data sent by a data owner and then concerns a certificate based on the encrypted message of this digital information. We use a group authentication protocol in the data generating structure. One of the organisations in the group produces valuable online information from raw data sent by a data owner and then concerns a certificate based on the encrypted message of this digital information.

#### Advantages:

- Accuracy is good and Low complexity.



## CHAPTER 5

### IMPLEMENTATION DETILES

#### CLOUD:

Cloud includes three basic services:

- Infrastructure as a Service (IaaS),
- Platform as a Service (PaaS), and
- Software as a Service (SaaS).

**Software-as-a-service (SaaS)** involves the licensure of a software application to customers. Licenses are typically provided through a pay-as-you-go model or on demand. This type of system can be found in Microsoft Office's 365

**Infrastructure-as-a-service (IaaS)** involves a method for delivering everything from operating systems to servers and storage through IP-based connectivity as part of an on-demand service. Clients can avoid the need to purchase software or servers, and instead procure these resources in an outsourced, on-demand service. Popular examples of the IaaS system include IBM Cloud and Microsoft Azure

**Platform-as-a-service (PaaS)** is considered the most complex of the three layers of cloud-based computing. PaaS shares some similarities with SaaS, the primary difference being that instead of delivering software online, it is actually a platform for creating software that is delivered via the Internet. This model includes platforms like Salesforce.com and Heroku.

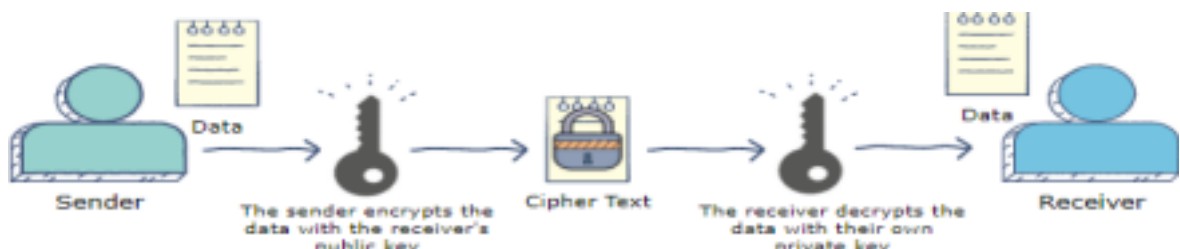


FIG 5.1 SENDER TO RECEIVER

## **5.1 DATA ENCRYPTION:**

Data encryption translates data into another form, or code, so that only people with access to a secret key (formally called a decryption key) or password can read it. Encrypted data is commonly referred to as ciphertext, while unencrypted data is called plaintext. Currently, encryption is one of the most popular and effective data security methods used by organizations. Two main types of data encryption exist - asymmetric encryption, also known as public-key encryption, and symmetric encryption.

### **PURPOSE:**

The purpose of data encryption is to protect digital data confidentiality as it is stored on computer systems and transmitted using the internet or other computer networks. The outdated data encryption standard (DES) has been replaced by modern encryption algorithms that play a critical role in the security of IT systems and communications.

These algorithms provide confidentiality and drive key security initiatives including authentication, integrity, and non-repudiation. Authentication allows for the verification of a message's origin, and integrity provides proof that a message's contents have not changed since it was sent. Additionally, non-repudiation ensures that a message sender cannot deny sending the message.

### **DATA DECRYPTION**

Decryption is the process of transforming data that has been rendered unreadable through encryption back to its unencrypted form. In decryption, the system extracts and converts the garbled data and transforms it to texts and images that are easily understandable not only by the reader but also by the system. Decryption may be accomplished manually or automatically. It may also be performed with a set of keys or passwords. One of the foremost reasons for implementing an encryption decryption system is privacy. As information travels over the World Wide Web, it becomes subject to scrutiny and access from unauthorized individuals or organizations. As a result, data is encrypted to reduce data loss and theft. Some of

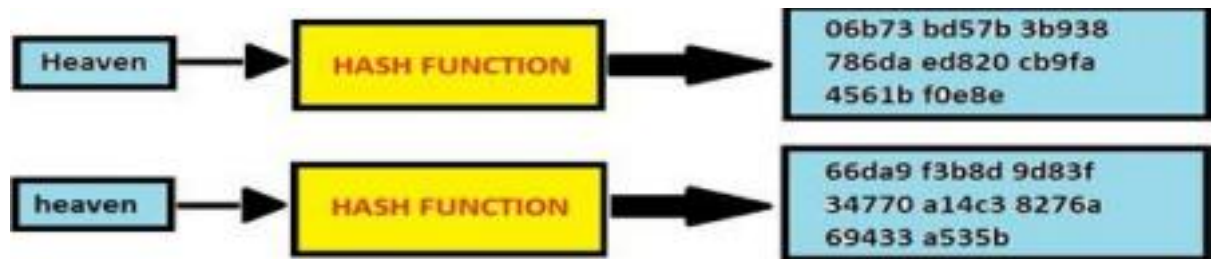
the common items that are encrypted include email messages, text files, images, user data and directories.

## **BLOCK CHAIN**

Blockchain is a system of recording information in a way that makes it difficult or impossible to change, hack, or cheat the system. A blockchain is essentially a digital ledger of transactions that is duplicated and distributed across the entire network of computer systems on the blockchain. Each block in the chain contains a number of transactions, and every time a new transaction occurs on the blockchain, a record of that transaction is added to every participant's ledger. The decentralised database managed by multiple participants is known as Distributed Ledger Technology (DLT).Blockchain is a type of DLT in which transactions are recorded with an immutable cryptographic signature called a hash. This means if one block in one chain was changed, it would be immediately apparent it had been tampered with. If hackers wanted to corrupt a blockchain system, they would have to change every block in the chain, across all of the distributed versions of the chain.

### **5.2 SHA Algorithm:**

SHA stands for secure hashing algorithm. SHA is a modified version of MD5 and used for hashing data and certificates. A hashing algorithm shortens the input data into a smaller form that cannot be understood by using bitwise operations, modular additions, and compression functions. You may be wondering, can hashing be cracked or decrypted? Hashing is similar to encryption, the only difference between hashing and encryption is that hashing is one-way, meaning once the data is hashed, the resulting hash digest cannot be cracked, unless a brute force attack is used. See the image below for the working of SHA algorithm. SHA works in such a way even if a single character of the message changed, then it will generate a different hash. For example, hashing of two similar, but different messages i.e., Heaven and heaven is different. However, there is only a difference of a capital and small letter.



**FIG 5.2 HASH FUNCTION**

### **IMPLEMENTATION:**

Data owner is the person who login's into the system after registration and upload his/her files into the cloud and view that file is secured or not .Data owner can view his files content using hash codes and if there is any improper content in that file, he can delete the complete file and upload the file with proper content.

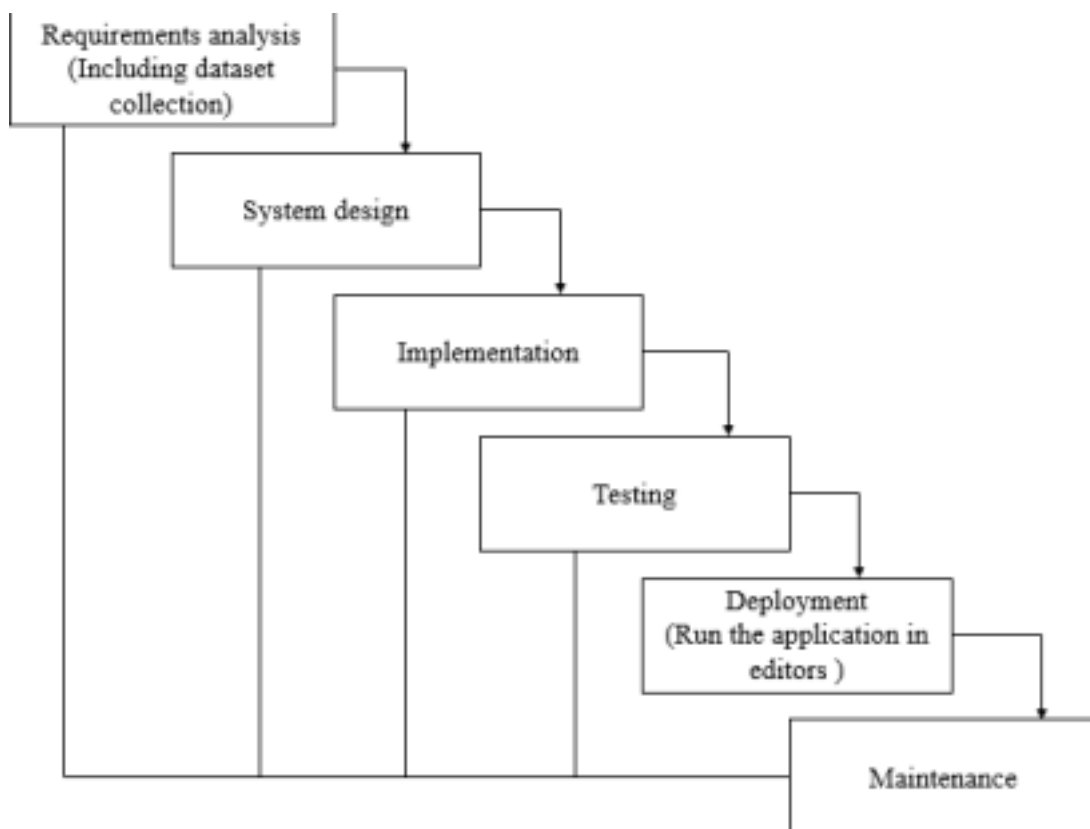
Data provider is the person who gives the protection to the data owner's uploaded files by using encryption technique and view his/her secured files and view user requests for files if is their any requests from users he can send keys to the userfor decryption. Data providers are added by the Group manager .

Group manager plays a vital role like adding data providers and views the data providers and finally logout from the system.

Data user is the person who can view all files which are uploaded by the data owner and sends request to the data provider if he accepts then the key will be generated for user and sent to the user for decrypting the encrypted data.This operations are performed by the data user after successful login and finally logout from the system.

## SOFTWARE DEVELOPMENT LIFE CYCLE – SDLC:

In our project we use waterfall model as our software development cycle because of its step-by-step procedure while implementing.



**FIG 5.3 WATERFALL MODEL**

- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

**Implementation** – With inputs from the system design, the system is first

developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

## **FEASIBILITY STUDY**

The feasibility of the project is analysed in this phase and 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. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis

- are ♦ **ECONOMICAL FEASIBILITY**
- ♦ **TECHNICAL FEASIBILITY**
- ♦ **SOCIAL FEASIBILITY**

### **Economic feasibility:**

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

.

### **Technical feasibility:**

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

### **Social feasibility:**

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

With the traditional data sharing method, the integrity of shared data is based on trust between the two partners participating in the exchange process. For example, doctors/hospitals absolutely believe that medical records received from their patients are integrity. In some cases, RO needs to ensure anonymity in MD generated by themselves. And the privacy of DO also needs to be protected as they don't want anyone to know which RO's service they used. In addition, the identities of those involved in the sharing process also need to be anonymous; and shared data need to be verified the reliability while still ensuring the privacy of its content.

### **Functional and non-functional requirements:**

Requirement's analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: Functional and non-functional requirements.

**Functional Requirements:** These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the

contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

Examples of functional requirements:

- 1) Authentication of user whenever he/she logs into the system
- 2) System shutdown in case of a cyber-attack
- 3) A verification email is sent to user whenever he/she register for the first time on some software system.

**Non-functional requirements:** These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

They basically deal with issues like:

- Portability
- Security
- Maintainability
- Reliability
- Scalability
- Performance
- Reusability
- Flexibility

Examples of non-functional requirements:

- 1) Emails should be sent with a latency of no greater than 12 hours from  
such an activity
- 2) The processing of each request should be done within 10 seconds
- 3) The site should load in 3 seconds whenever of simultaneous users are > 10000

### 5.3 SYSTEM SPECIFICATIONS:



## H/W SPECIFICATIONS:

- **Processor - I3/Intel Processor · RAM - 8GB (min)**
- Hard Disk - 128 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - Any

## S/W SPECIFICATIONS:

- Operating System : Windows 10
- Server-side Script : Python 3.6
- IDE : PyCharm
- Libraries Used : Pandas, MYSQL
- Framework : Flask.

## SYSTEM DESIGN

### Input Design:

In an information system, input is the raw data that is processed to produce output. During the input design, the developers must consider the input devices such as PC, MICR, OMR, etc.

Therefore, the quality of system input determines the quality of system output.

Well-designed input forms and screens have following properties –

- It should serve specific purpose effectively such as storing, recording, and  
17  
retrieving the information.
- It ensures proper completion with accuracy.
- It should be easy to fill and straightforward
- It should focus on user's attention, consistency, and simplicity

All these objectives are obtained using the knowledge of basic design principles regarding

- What are the inputs needed for the system?

- o How end users respond to different elements of forms and screens.

### **Objectives for Input Design:**

The objectives of input design are –

- To design data entry and input procedures
- To reduce input volume
  - To design source documents for data capture or devise other data capture methods
- To design input data records, data entry screens, user interface screens, etc.
- To use validation checks and develop effective input controls.

### **Output Design:**

The design of output is the most important task of any system. During output design, developers identify the type of outputs needed, and consider the necessary output controls and prototype report layouts.

### **Objectives of Output Design:**

The objectives of input design are:

- To develop output design that serves the intended purpose and eliminates the production of unwanted output.
- To develop the output design that meets the end user's requirements.

To deliver the appropriate quantity of output.

18

- To form the output in appropriate format and direct it to the right person.
- To make the output available on time for making good decisions.

### **MODULES:**

#### **1. DataOwner:**

**Login:** Dataowner has to login with valid details which are used in his / her Registration

**Register:** Each and every Dataowner has to Register.

**Upload files:** uploads his files into the cloud.

**View Files :** views all the files which are uploaded by the him/her and sends request if there is any requests.

**Secured Files:** He can view his/her files are secured or not.

**Logout :**Finally logout from the system.

## 2. Data Provider:

**Login:** data provider has to login with valid details which are used in his / her Registration

**Register:** Each and every data provider has to Register and management has to accept request..

**View Data Owners Request :** Views data owners requests and secure those files.

**View Secured Files:** data provider can view all files which are secured.

**User Requests :** views user requests for files and sends keys to the users.

**Logout :**Finally logout from the system.

## 3. Group Manager:

**Login:** Manager will login with default details

**Add providers:** He adds the data providers and views all the data providers who are added by him

**Logout:** Finally logout from the system.

19

## 4. Data User:

**Register:** User registers with details

**Login:** Datauser will login with his credentials which are added in his/her registration.

**View Files :** He views all the files which are uploaded by the data provider. **Send Request:** sends the request for file

**MyFiles:** views all the files which are accepted

**Check Files:** he need hash codes to view the file content

**Download file:** if hash codes are valid the data will be decrypted and

can download as txt file

**Logout:** Finally logout from the system.

## 5.4 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

20

### **GOALS:**

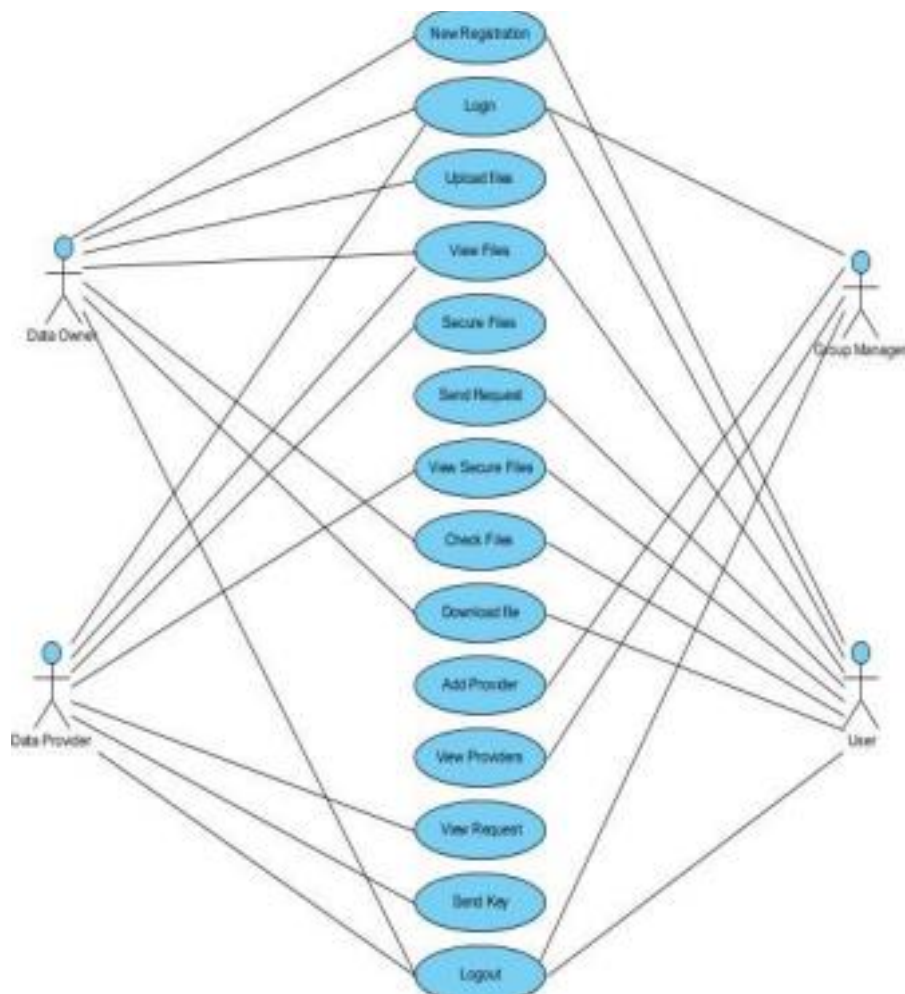
The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.

6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

## **USE CASE DIAGRAM**

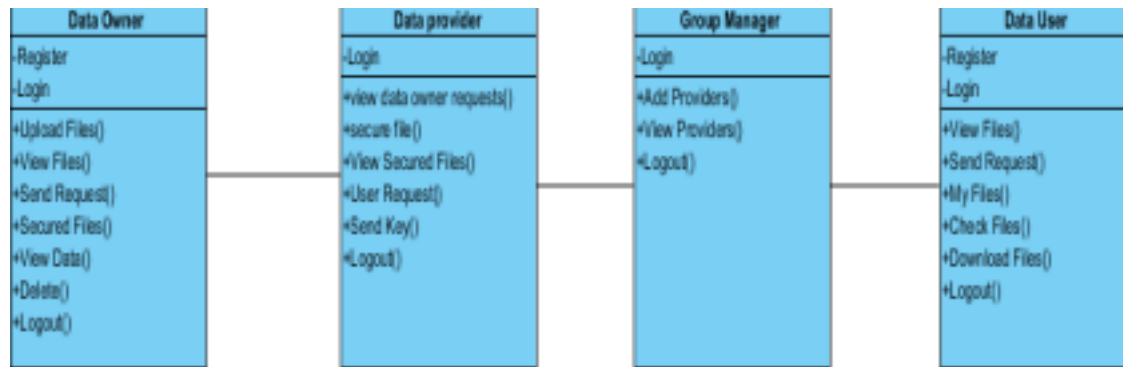
- ▶ A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis.
- ▶ Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.
- ▶ The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



**FIG 5.4 USECASE DIAGRAM**

## CLASS DIAGRAM

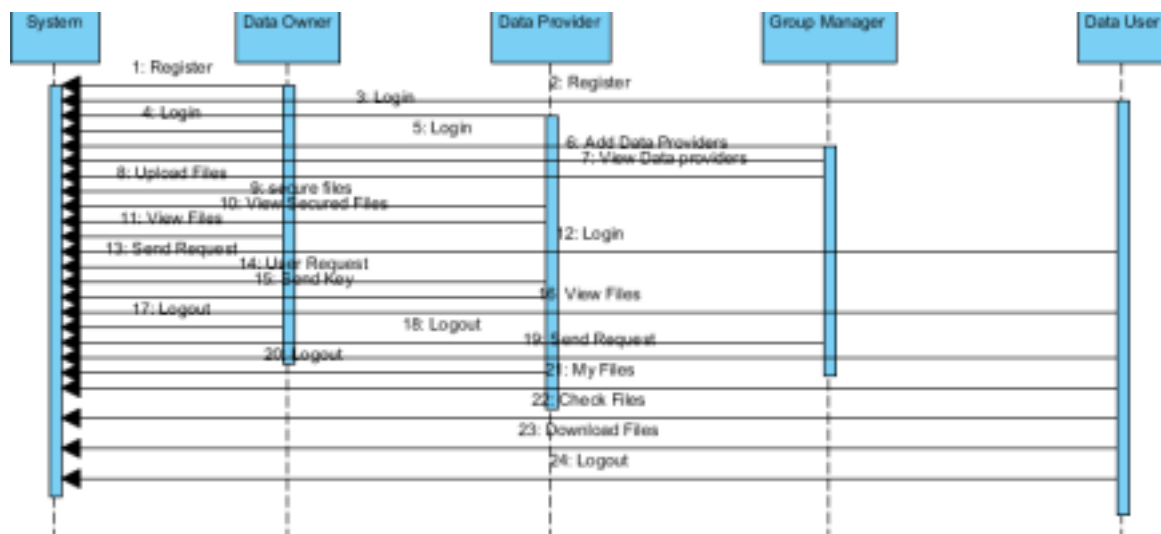
In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**FIG 5.5 CLASS DIAGRAM**

## SEQUENCE DIAGRAM

- A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order.
- It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

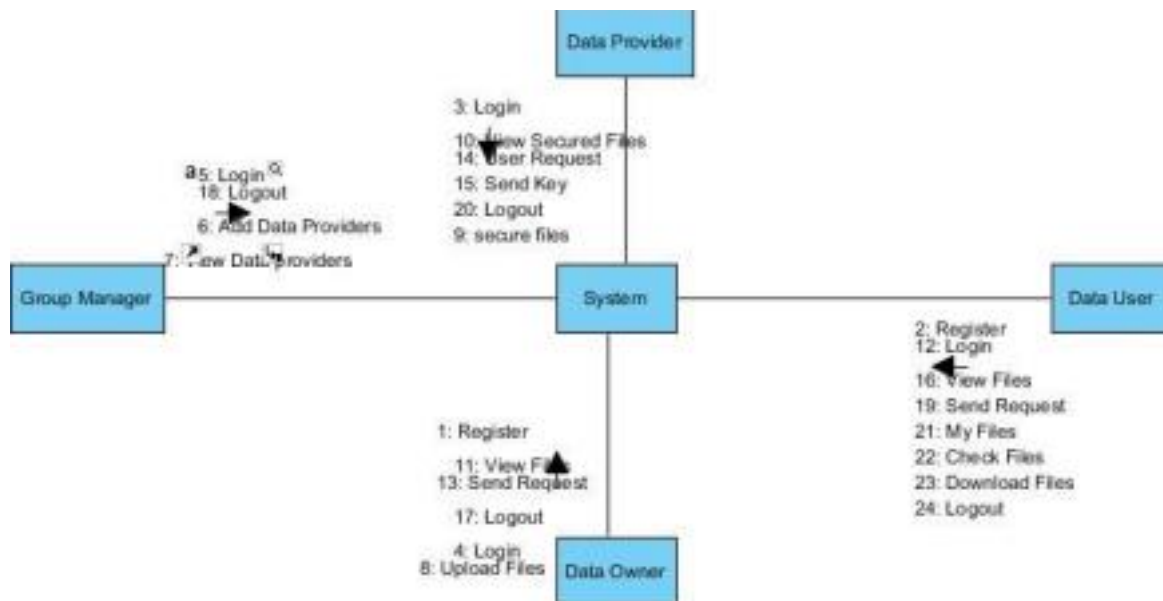


**FIG 5.6 SEQUENCE DIAGRAM**

## COLLABORATION DIAGRAM:

In collaboration diagram the method call sequence is indicated by some numbering technique as shown below. The number indicates how the methods are called one

after another. We have taken the same order management system to describe the collaboration diagram. The method calls are similar to that of a sequence diagram. But the difference is that the sequence diagram does not describe the object organization whereas the collaboration diagram shows the object organization.



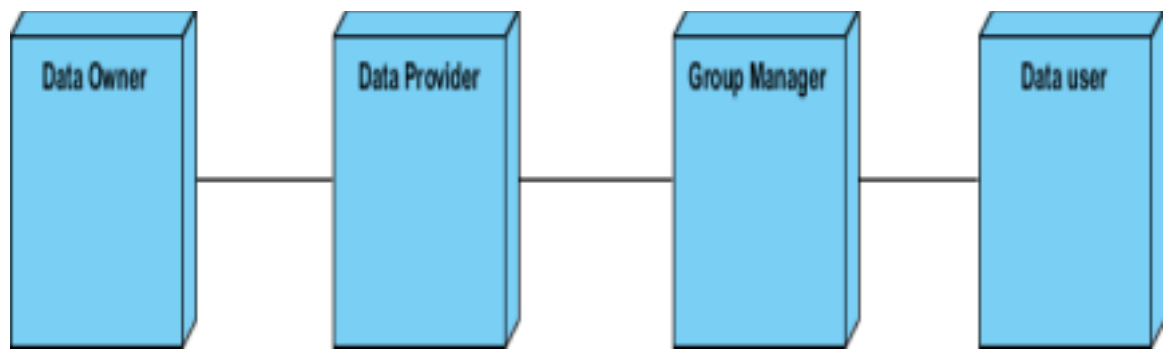
**FIG 5.7 COLLABORATION DIAGRAM**

## DEPLOYMENT DIAGRAM

Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware's used to deploy the application. With the traditional data sharing method, the integrity of shared data is based on trust between the two partners participating in the exchange process. For example, doctors/hospitals absolutely believe that medical records received from their patients are integrity. In some cases, RO needs to ensure anonymity in MD generated by themselves. And the privacy of DO also needs to be protected as they don't want anyone to know which RO's service they used. In addition, the identities of those involved in the

sharing process also need to be anonymous; and shared data need to be verified the reliability while still ensuring the privacy of its content.





**FIG 5.8 DEPLOYMENT DIAGRAM**

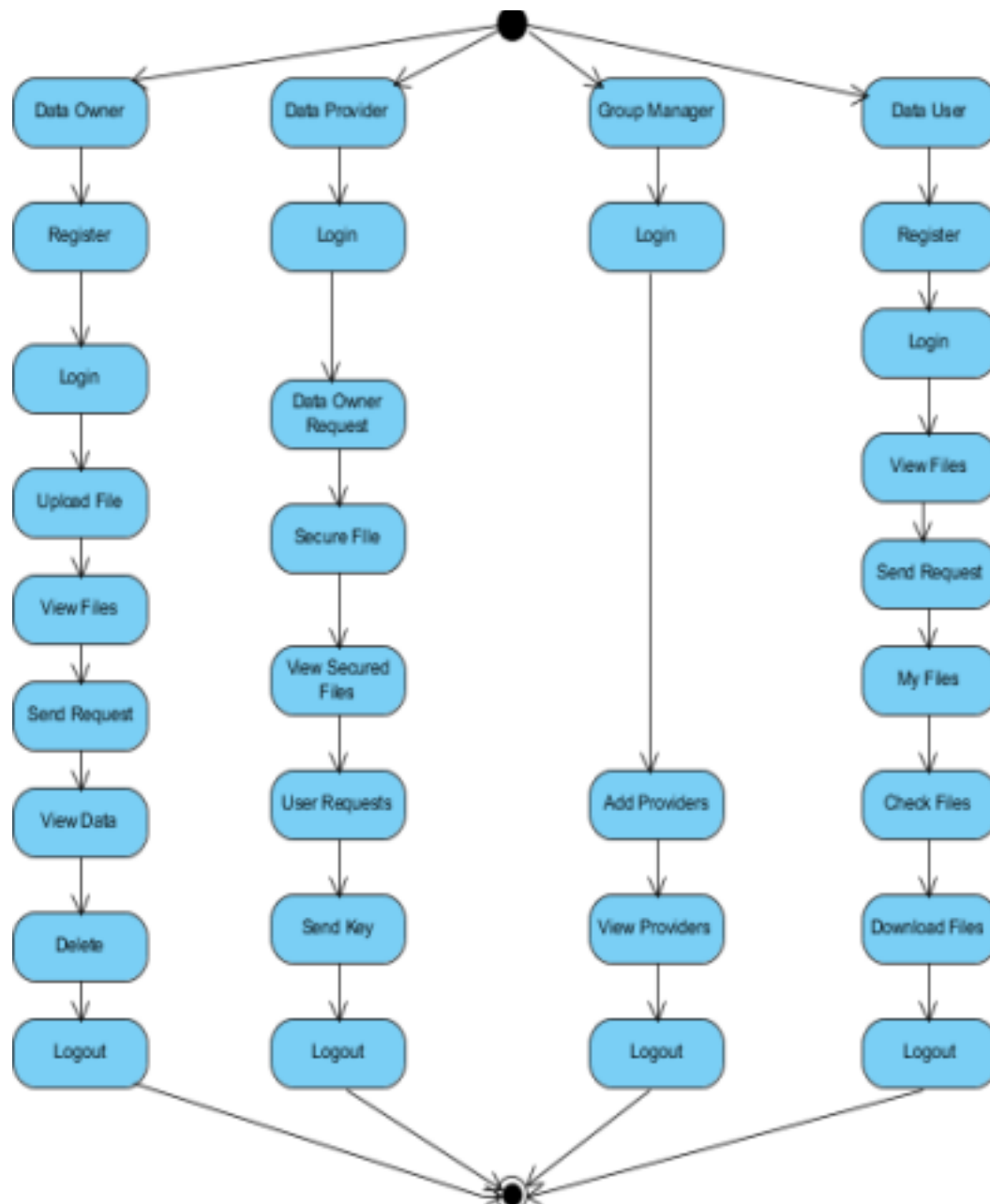
### **ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

You can make an activity diagram by connecting and joining various activity states. The starting point is usually marked with a dark, filled-in circle with an arrow pointing to the next state usually a rectangle with rounded corners. All action flows are represented with arrows indicating the transitions from state to state. Activity diagrams present a number of benefits to users. Consider creating an activity

diagram to:

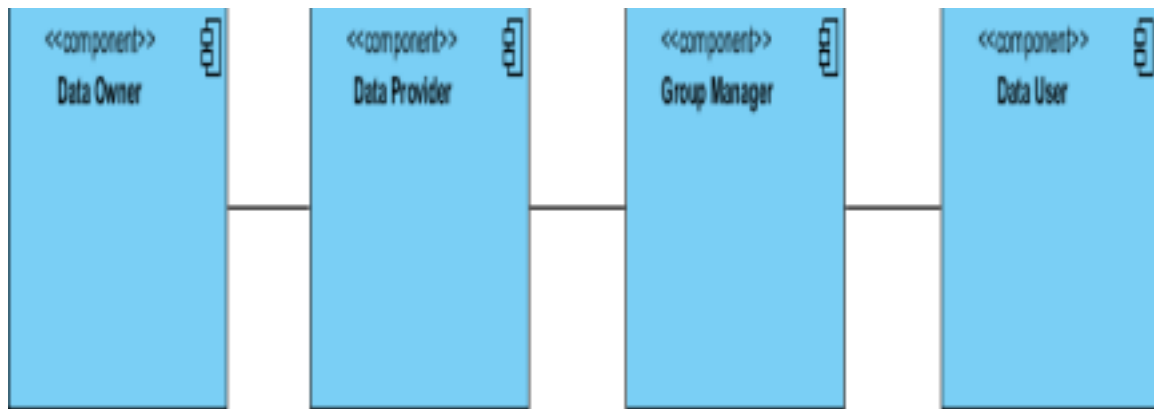
- Demonstrate the logic of an algorithm.
- Describe the steps performed in a UML use case.
- Illustrate a business process or workflow between users and the system.



**FIG 5.9 ACTIVITY DIAGRAM**

#### **COMPONENT DIAGRAM:**

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development.

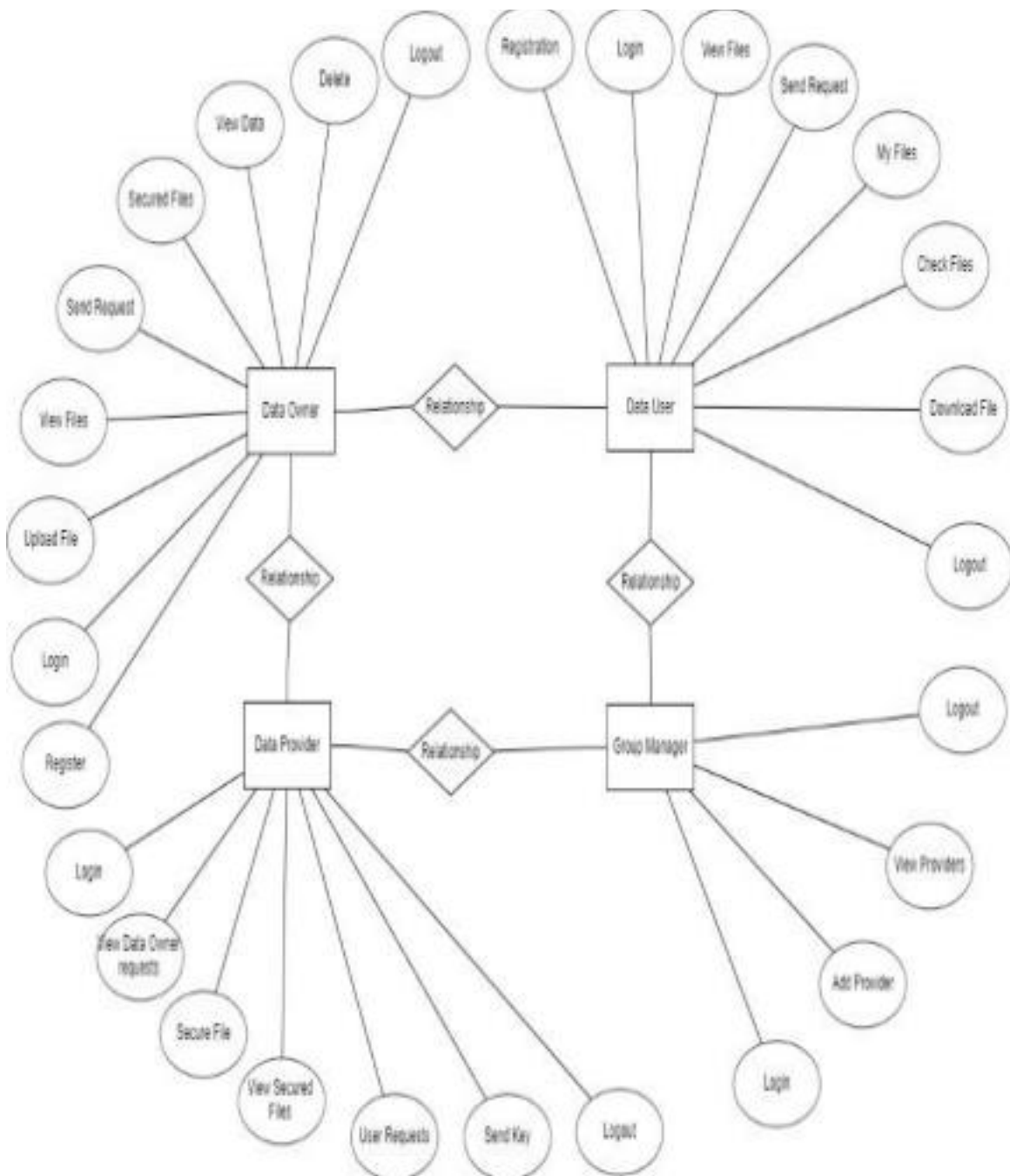


**FIG 5.10 COMPONENT DIAGRAM**

### **ER DIAGRAM:**

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Let's have a look at a simple ER diagram to understand this concept.

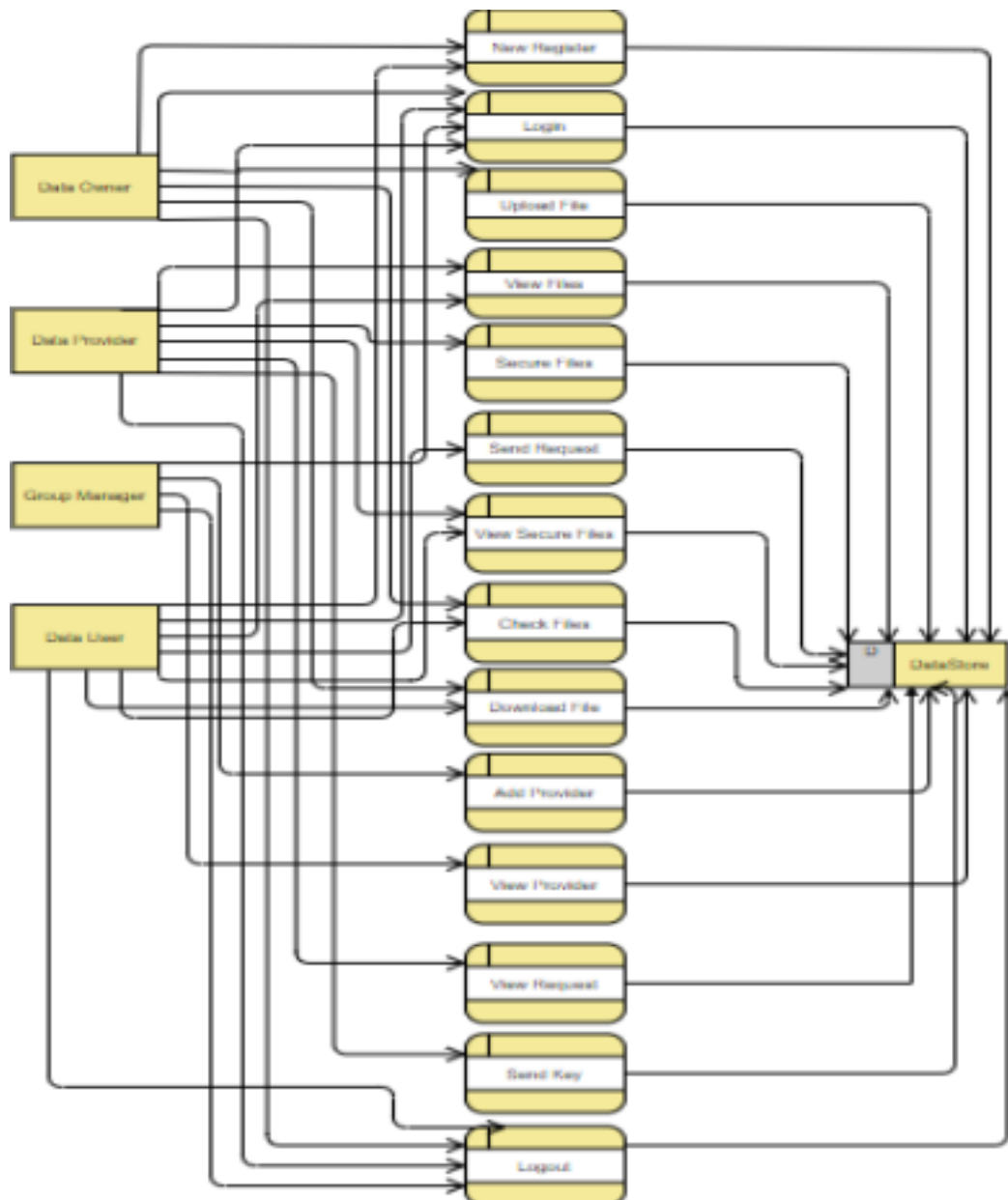


**FIG 5.11 ER DIAGRAM**

### DFD DIAGRAM:

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It

shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a



**FIG 5.12 DFD DIAGRAM**

part in the system that acts as the starting point for redesigning a system.

**FIG 5.13 DFD DIAGRAM LEVEL 1**

sharing. In the data producing scheme, we consider RO as DP, a group manager sets up a group of DPs providing the same type of services. DP can generate MD from RD sent from DO, and then issues a certificate on EMD. In the data storing scheme, we provide not only the confidentiality and integrity of the stored data but also the anonymity of DP and the privacy of DO which have not been fulfilled in the existing solutions. In the data sharing scheme, everyone on the system can verify the reliability of shared data before submitting a sharing request to DO. Note that everyone can only verify the reliability of the shared data but cannot read its contents. This property could not be fulfilled by existing solutions. In addition, the data sharing process is done directly between DO and DU without depending on any intermediaries. The results of the security analysis show that the proposed schemes meet the security properties including confidentiality , integrity, privacy, non-repudiation, and anonymity.

In this paper, we propose three schemes: data producing, data storing, and data sharing. In the data producing scheme, we consider RO as DP, a group manager

sets up a group of DPs providing the same type of services. DP can generate MD from RD sent from DO, and then issues a certificate on EMD. In the data storing scheme, we provide not only the confidentiality and integrity of the stored data but also the anonymity of DP and the privacy of DO which have not been fulfilled in the existing solutions. In the data sharing scheme, everyone on the system can verify the reliability of shared data before submitting a sharing request to DO. Note that everyone can only verify the reliability of the shared data but cannot read its contents. This property could not be fulfilled by existing solutions. In addition, the data sharing process is done directly between DO and DU without depending on any intermediaries. The results of the security analysis show that the proposed schemes meet the security properties including confidentiality , integrity, privacy, non-repudiation, and anonymity.

## **FUTURE SCOPE**

In our future work, we will apply the proposed system to specific applications such as IoT, electronic medical records. We will then evaluate and optimize the schemes

## **REFERENCES:**

- [1] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," IDC White Paper, Nov. 2018.
- [2] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *EEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 909–928, 2nd Quart., 2013.



[3] L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An IoT-oriented data storage framework in cloud computing platform," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1443–1451, May 2014.

[4] T. A. Phan, J. K. Nurminen, and M. Di Francesco, "Cloud databases for Internet-of-Things data," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom)*, Sep. 2014, pp. 117–124.

[5] K. Yasumoto, H. Yamaguchi, and H. Shigeno, "Survey of real-time processing technologies of IoT data streams," *J. Inf. Process.*, vol. 24, no. 2, pp. 195–202, 2016.

[6] A. Kumar, N. C. Narendra, and U. Bellur, "Uploading and replicating Internet of Things (IoT) data on distributed cloud storage," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 670–677.

[7] K. Hossain, M. Rahman, and S. Roy, "IoT data compression and optimization techniques in cloud storage: Current prospects and future directions," *Int. J. Cloud Appl. Comput.*, vol. 9, no. 2, pp. 43–59, Apr. 2019.

[8] J. D. Bokefode, A. S. Bhise, P. A. Satarkar, and D. G. Modani, "Developing a secure cloud storage system for storing IoT data by applying role based encryption," *Procedia Comput. Sci.*, vol. 89, no. 1, pp. 43–50, 2016.

33

[9] W. Wang, P. Xu, and L. T. Yang, "Secure data collection, storage and access in cloud-assisted IoT," *IEEE Cloud Comput.*, vol. 5, no. 4, pp. 77–88, Jul. 2018.

[10] M. Rashid, S. A. Parah, A. R. Wani, and S. K. Gupta, "Securing Ehealth IoT data on cloud systems using novel extended role based access control model," in *Internet Things (IoT)*. Cham, Switzerland: Springer, 2020, pp. 473–489.

[11] R. Arora, A. Parashar, and C. C. I. Transforming, "Secure user data in cloud computing using encryption algorithms," *Int. J. Eng. Res. Appl.*, vol. 3, no. 4, pp. 1922–1926, 2013.

[12] M. Kantarcioglu and F. Shaon, "Securing big data in the age of AI," in *Proc. 1st IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl. (TPSISA)*, Dec. 2019, pp.

[13] C. Zhang, J. Sun, X. Zhu, and Y. Fang, “Privacy and security for online social networks: Challenges and opportunities,” *IEEE Netw.*, vol. 24, no. 4, pp. 13–18, Jul./Aug. 2010.

[14] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. Sebastopol, CA, USA: O’Reilly Media, 2014.

[15] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, “Blockchain challenges and opportunities: A survey,” *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

[16] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, “A survey on security and privacy issues of bitcoin,” *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018.

[17] Q. Xia, E. B. Sifah, A. Smahi, S. Amofa, and X. Zhang, “BBDS:Blockchain based data sharing for electronic medical records in cloud environments,” *Information*, vol. 8, no. 2, p. 44, 2017.

## APPENDIX

### A. SOURCE CODE:

```
from flask import
Flask,render_template,request,session,redirect,url_for from datetime
import datetime
from datetime import date
import pandas as pd
import os
import hashlib

import smtplib
```

```

from flask_mail import *

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText


import mysql.connector

mydb=mysql.connector.connect(user='root',password='',port=3306,database='rgs
d ss')

cur=mydb.cursor()

app=Flask(__name__)


app.config['UPLOAD_FOLDER']

="D:/progress/Datastoringandsharing/upload/"

app.secret_key="%7868chbhyvd6sb6sjdcbsdycgdbc"


@app.route('/')


def index():
    35
    return render_template('index.html')


@app.route('/data_ownerreg',methods=["POST","GET"])

def dataownerreg():

    if request.method=='POST':

        name=request.form['name']

        email=request.form['email']

        password=request.form['password']

        confirmpassword=request.form['confirmpassword']

        contact=request.form['contact']

```

```

address=request.form['address']

if password == confirmpassword:

    sql = "select * from dataownerregistration where email='%s'

    '%(email) cur.execute(sql)

    data=cur.fetchall()

    mydb.commit()

    if data == []:

        sql="insert into dataownerregistration
(name,email,password,confirmpassword,contact,address) values
('%s','%s','%s','%s','%s','%s')"%(name,email,password,confirmpassword,contact,a
ddress)

        cur.execute(sql)

        mydb.commit()

        return render_template('dataownerlogin.html')

        msg="credential already exists with this Email"

        36
        return render_template('dataownerreg.html', msg=msg)
        msg="Password Didn't match"

        return render_template('dataownerreg.html',msg=msg)

return render_template('dataownerreg.html')

@app.route('/dataowner_login',methods=['POST','GET'])

def dataownerlogin():

    if request.method=='POST':

        name=request.form['name']

        session['dataowner']=name

        password=request.form['password']

        session['password'] = password

        print(session['dataowner'],session['password'])

```

```
sql = "select * from dataownerregistration where name='%s' and  
password='%s'" % (session['dataowner'],session['password'])
```

```
cur.execute(sql)
```

```
data = cur.fetchall()
```

```
mydb.commit()
```

```
print(data)
```

```
if data == []:
```

```
    msg="Credentials are not Valid"
```

```
    return render_template('dataownerlogin.html',msg=msg)
```

```
    return render_template('dataownerhome.html',name=name)
```

```
    return render_template('dataownerlogin.html')
```

```
@app.route('/Upload_file',methods=['POST','GET'])
```

37

```
def uploadfile():
```

```
    if request.method=='POST':
```

```
        filedata=request.files['file']
```

```
        filename=filedata.filename
```

```
        file_data=filedata.read()
```

```
        print(file_data)
```

```
        path=os.path.join("upload/",filename)
```

```
        filedata.save(path)
```

```
        today = date.today()
```

```
        d1 = today.strftime("%d/%m/%Y")
```

```
        timining=datetime.now()
```

```
        ct=timining.strftime("%H:%M:%S")
```

```
        sql="select * from dataowneruploadfiles where FileOwner='%s' and  
Filename='%s'"%(session['dataowner'],filename)
```

```
cur.execute(sql)
```

```
data=cur.fetchall()
```

```
mydb.commit()
```

```
if data == []:
```

```
    sql="insert into  
dataowneruploadfiles(Filename,Filedata,Fileowner,Date,Time)values(%s,%s,%s,%  
s,%s)"
```

```
    val=(filename,file_data,session['dataowner'],d1,ct)
```

```
    # sql="insert into  
dataowneruploadfiles(Filename,Filedata,Fileowner,Date,Time)values('%s','%s','%s'  
, '%s', '%s')"% (filename,file_data,session['dataowner'],d1,ct)
```

```
    cur.execute(sql,val)
```

38

```
mydb.commit()
```

```
msg="File uploaded successfully"
```

```
return render_template('uploadfile.html',msg=msg)
```

```
msg = "File already exists"
```

```
return render_template('uploadfile.html',msg=msg)
```

```
return render_template('uploadfile.html')
```

```
@app.route('/viewownerfiles')
```

```
def viewownerfiles():
```

```
    sql="select SIno,Filename,Date,Time from dataowneruploadfiles where  
FileOwner='%s'"%(session['dataowner'])
```

```
    data=pd.read_sql_query(sql,mydb)
```

```
    sql = "select name from dataproviders"
```

```
    cur.execute(sql)
```

```
    data1=cur.fetchall()
```

```
    data1=[j for i in data1 for j in i]
```

```

        return
    render_template('viewownerfiles.html',cols=data.columns.values,rows=data.values.tolist(),data1=data1)

```

```

@app.route('/combinedata/<z>',methods=['POST','GET'])

```

```

def combinedata(z=0):

```

```

    sql = "select Slno,Filename,Date,Time from dataowneruploadfiles where
    FileOwner='%s' and status='pending'" % (session['dataowner'])

```

```

    data = pd.read_sql_query(sql, mydb)

```

```

    if request.method=='POST':

```

```

        names=request.form['dataprovder']

```

39

```

        sql="update dataowneruploadfiles set Dataprovder='%s',status='requested'
        where Slno='%s'"%(names,z)

```

```

        cur.execute(sql)

```

```

        mydb.commit()

```

```

        sql="select Slno,Filename,Dataprovder from dataowneruploadfiles where
        status='requested' and FileOwner='%s'"%(session['dataowner'])

```

```

        data1 = pd.read_sql_query(sql, mydb)

```

```

        msg="File Request sent Successfully"

```

```

        return

```

```

    render_template('combinedata.html',col=data1.columns.values,row=data1.values.tolist(),msg=msg)

```

```

    return

```

```

    render_template('viewownerfiles.html',cols=data.columns.values,rows=data.values.tolist())

```

```

@app.route('/upprovider/<x>')

```

```

def upprovider(x=""):

```

```

    sql="update dataowneruploadfiles set status='accepted' where

```

```
status='requested' and SIno='%s'"%(x)
```

```
cur.execute(sql)
```

```
mydb.commit()
```

```
return redirect(url_for('viewownerfiles'))
```

```
@app.route('/securedfiles')
```

```
def securedfiles():
```

```
    sql="select          SIno,Filename,Hash1,Hash2,Dataprovider          from  
dataowneruploadfiles  where      status='allow'      and      FileOwner='%s'  
"%(session['dataowner'])
```

```
40
```

```
data=pd.read_sql_query(sql,mydb)
```

```
    return  
render_template('securedfiles.html',cols=data.columns.values,rows=data.values.t  
olist())
```

```
@app.route('/ownercheckfiles/<u>/<v>/')
```

```
def ownercheckfiles(u="",v="):
```

```
    print(u)
```

```
    print(v)
```

```
    sql = "select AES_DECRYPT(Fileencdata, 'key') from dataowneruploadfiles  
where Hash1='%s' and Hash2='%s' " % (u,v)
```

```
cur.execute(sql)
```

```
data = cur.fetchall()
```

```
mydb.commit()
```

```
data = [j for i in data for j in i]
```

```
data=data[0].decode()
```

```
return render_template('ownercheckfiles.html',data=data)
```



```
@app.route('/deletefile/<d>')
```

```
def deletefile(d=0):
```

```
    sql="delete from dataowneruploadfiles where SIno='%s'"%(d)
```

```
    cur.execute(sql)
```

```
    mydb.commit()
```

```
    return redirect(url_for('securedfiles'))
```

41

```
@app.route('/ownerlogout')
```

```
def ownerlogout():
```

```
    session.pop('dataowner',None)
```

```
    return redirect(url_for('index'))
```

```
@app.route('/Group_manager',methods=['POST','GET'])
```

```
def Groupmanager():
```

```
    if request.method=='POST':
```

```
        name=request.form['name']
```

```
        password=request.form['password']
```

```
        session['manager']=name
```

```
        if name=='Manager' and password=='group':
```

```
            return render_template('managerhome.html',name=name)
```

```
        else:
```

```
            msg="Invalid Details"
```

```
            return render_template('Groupmanager.html',msg=msg)
```

```
    return render_template('Groupmanager.html')
```

```
@app.route('/allproviders')
```

```
def allproviders():
```

```
    sql="select name,email,contact,address from dataproviders"
```

```

data=pd.read_sql_query(sql,mydb)

mydb.commit()

return
render_template('allproviders.html',cols=data.columns.values,rows=data.values.tolist())

```

42

```

@app.route('/managerlogout')

def managerlogout():

    session.pop('manager',None)

    return redirect(url_for('index'))

@app.route('/data_providers',methods=['POST','GET'])

def dataproviders():

    if request.method=='POST':

        name=request.form['name']

        Email=request.form['email']

        password=request.form['password']

        confirmpassword=request.form['confirmpassword']

        contact = request.form['contact']

        address = request.form['address']

        if password == confirmpassword:

            sql="select * from dataproviders where name='%s' and email='%s'"%(name,Email)

            cur.execute(sql)

            data=cur.fetchall()

            mydb.commit()

            if data == []:

                sql="insert into dataproviders (name,email,password,confirmpassword,contact,address) values ('%s','%s','%s','%s','%s','%s')"%(name,Email,password,confirmpassword,contact,a

```

ddress)

```
cur.execute(sql)
```

```
mydb.commit()
```

```
content = "Credentials to login RGS & DSS "
```

43

```
mail_content = content
```

```
sender_address = 'madhanvenkatareddy596@gmail.com'
```

```
sender_pass = 'kbfscpeycmqaknXP'
```

```
receiver_address = Email
```

```
message = MIMEMultipart()
```

```
message['From'] = sender_address
```

```
message['To'] = receiver_address
```

```
message['Subject'] = 'a reliability guaranteed solution for data storing  
and sharing'
```

```
message.attach(MIMEText(mail_content, 'plain'))
```

```
section = smtplib.SMTP('smtp.gmail.com', 587)
```

```
section.starttls()
```

```
section.login(sender_address, sender_pass)
```

```
text = message.as_string()
```

```
section.sendmail(sender_address, receiver_address, text)
```

```
section.quit()
```

```
msg="Details added successfully"
```

```
return render_template('dataproviders.html', msg=msg)
```

```
msg="Details already exists"
```

```
return render_template('dataproviders.html',msg=msg)
```

```
return render_template('dataproviders.html')
```

```
@app.route('/dataproviderslogin',methods=['POST','GET'])
```

```

def dataproviderslogin():

    if request.method=='POST':

        name=request.form['name']

        session['provider']=name

        password=request.form['password']

        session['password']=password

        sql = "select * from dataproviders where name='%s' and password='%s'" %
(session['provider'], session['password'])

        cur.execute(sql)

        data = cur.fetchall()

        mydb.commit()

        if data==[]:

            msg="Credentials are not valid please check your mail and re-enter
username and password"

            return render_template('dataproviderslogin.html',msg=msg)

        return

    render_template('dataprovidershome.html',name=name) return

    render_template('dataproviderslogin.html')

@app.route('/dataownerrequests')

def dataownerrequests():

    sql="select SIno,FileOwner,Filename from dataowneruploadfiles where
Dataprovider='%s' and status='accepted'"%(session['provider'])

    data=pd.read_sql_query(sql,mydb)

    return

    render_template('dataownerrequests.html',cols=data.columns.values,rows=data.v
a lues.tolist())

```

```
@app.route('/securefile/<r1>/<r2>')
```

```
def securefile(r1=0,r2="):
```

```
    sql="select Filedata from dataowneruploadfiles where
```

```
    Sln0='%s'%(r1) cur.execute(sql)
```

```
    data=cur.fetchall()
```

```
    mydb.commit()
```

```
    enc_data=data[0][0]
```

```
    datalen = int(len(enc_data) / 2)
```

```
    g = 0
```

```
    a = "
```

```
    b = "
```

```
    c = "
```

```
    for i in range(0, 2):
```

```
        if i == 0:
```

```
            a = enc_data[g: datalen:1]
```

```
            # a=a.decode('utf-8')
```

```
            result = hashlib.sha1(a.encode())
```

```
            hash1 = result.hexdigest()
```

```
        c = enc_data[datalen: len(enc_data):1]
```

```
        # c = c.decode('utf-8')
```

```
        result = hashlib.sha1(c.encode())
```

```
        hash2 = result.hexdigest()
```

```

    sql="update dataowneruploadfiles set
FileEncData=AES_ENCRYPT('%s','key'),Dataone=AES_ENCRYPT('%s','key'),Dat
atwo=AES_ENCRYPT('%s','key'),status='allow',Hash1='%s',Hash2='%s' where
Sln='%s' and status='accepted'""%(enc_data,a,c,hash1,hash2,r1)

```

```

    cur.execute(sql)

```

```

    mydb.commit()

```

```

    # path="D:/progress/Datastoringandsharing/upload/",r2

```

```

    # isfile=os.path.isfile(path)

```

```

    # print(isfile)

```

```

    # sql="select Filename from dataowneruploadfiles where

```

```

    Sln='%s'""%(r1) # cur.execute(sql)

```

```

    # data=cur.fetchall()

```

```

    # mydb.commit()

```

```

    # data=[j for i in data for j in i]

```

```

    # print(data[0])

```

```

    return redirect(url_for('dataownerrequests'))

```

```

@app.route('/providersecurefiles')

```

```

def providersecurefiles():

```

```

    sql="select Sln,Filename,FileOwner from dataowneruploadfiles where
Dataprovider='%s' and status='allow'""%(session['provider'])

```

```

    data=pd.read_sql_query(sql,mydb)

```

```

    print(data)

```

47

```

    return

```

```

    render_template('providersecurefiles.html',cols=data.columns.values,rows=data.v
a lues.tolist())

```

```

@app.route('/fromuser')

```

```

def fromuser():

```

```

    sql="select SIno,FileOwner,Filename,user from dataowneruploadfiles where
Action='accepted'"

    data=pd.read_sql_query(sql,mydb)

    return
render_template('fromuser.html',cols=data.columns.values,rows=data.values.tolist
( ))

```

```

@app.route('/sendrequest/<y>/<n>')

```

```

def sendrequest(n="",y=""):

```

```

    print(n)

```

```

    print(y)

```

```

    sql="select * from datauser where name='%s'"%(y)

```

```

    cur.execute(sql)

```

```

    data=cur.fetchall()

```

```

    mydb.commit()

```

```

    useremail=data[0][2]

```

```

    sql="select Hash1,Hash2 from dataowneruploadfiles where Filename='%s' and
user='%s'"%(n,y)

```

```

    cur.execute(sql)

```

```

    data=cur.fetchall()

```

```

    mydb.commit()

```

```

    print(data)

```

```

    key1=data[0][0]

```

48

```

    key2=data[0][1]

```

```

    print(type(key1))

```

```

    print(type(key2))

```

```

    content = f"The key to decrypt files :key 1 : {key1} key 2 : {key2} "

```

```

    print(content)

```

```

sender_address = 'madhanvenkatareddy596@gmail.com'

sender_pass = 'kbfscpeycmqaknpx'

receiver_address = useremail

message = MIMEMultipart()

message['From'] = sender_address

message['To'] = receiver_address

message['Subject'] = 'Reliability Guarenteed solution for Data Storing and Data
Sharing'

message.attach(MIMEText(content, 'plain'))

section = smtplib.SMTP('smtp.gmail.com', 587)

section.starttls()

section.login(sender_address, sender_pass)

text = message.as_string()

section.sendmail(sender_address, receiver_address, text)

section.quit()

sql="Update dataowneruploadfiles set Action='accept' where Action='accepted'
and Filename='%s' and user='%s'"%(n,y)

cur.execute(sql)

mydb.commit()

return redirect(url_for('fromuser'))

return render_template('sendrequest.html')

```

49

```

@app.route('/datapviderlogout')

def datapviderlogout():

    session.pop('provider',None)

    return redirect(url_for('index'))

@app.route('/datareg',methods=["POST","GET"])

```



```

def datareg():

    if request.method=="POST":

        name=request.form['name']

        email=request.form['email']

        password=request.form['password']

        confirmpassword=request.form['confirmpassword']

        contact=request.form['contact']

        address=request.form['address']

        if password==confirmpassword:

            sql="select * from datauser where name='%s' and
email='%s'"%(name,email)

            cur.execute(sql)

            data=cur.fetchall()

            mydb.commit()

            print(data)

            if data==[]:

                sql="insert into datauser(name,email,password,contact,address) values
('%s','%s','%s','%s','%s')"%(name,email,password,contact,address)

                cur.execute(sql)

                mydb.commit()

                return render_template('datauser.html')

                msg="Account Already Exists"

                return render_template('datareg.html',msg=msg)

            return render_template('datareg.html')

    @app.route('/datauser',methods=['POST','GET'])

def datauser():

```

```

if request.method=='POST':

    name=request.form['name']

    password=request.form['password']

    session['user']=name

    sql="select * from datauser where name='%s' and
password='%s'"%(session['user'],password)

    cur.execute(sql)

    data=cur.fetchall()

    mydb.commit()

    if data== []:

        msg="details are not valid"

        return render_template('datauser.html',msg=msg)

    return render_template('userhome.html')

return render_template('datauser.html')

```

51

```

@app.route('/userviewfiles')

def userviewfiles():

    sql="select SIno,FileOwner,Filename,Fileencdata,Dataprovider from
dataowneruploadfiles where status='allow' and Action='pending' "

    data=pd.read_sql_query(sql,mydb)

    msg="request sent successfully"

    return
render_template('userviewfiles.html',cols=data.columns.values,rows=data.values
.t olist(),msg=msg)

@app.route('/userrequest/<x>/<y>')

def userrequest(x=0,y=""):

```

```

print(x,y)

sql="select SIno,Filename,Fileencdata from dataowneruploadfiles where
SIno='%s' and Filename='%s'"%(x,y)

cur.execute(sql)

data=cur.fetchall()

mydb.commit()

print(data)

fileid=data[0][0]

Filename=y

encdata=data[0][2]

# sql="insert into userfiles(Fileid,Filename) values('%s','%s')

"%(fileid,Filename) # cur.execute(sql)

# mydb.commit()

sql="update dataowneruploadfiles set user='%s',Action='accepted' where
SIno='%s' and Filename='%s'"%(session['user'],x,y)

52

cur.execute(sql)

mydb.commit()

return redirect(url_for('userviewfiles'))

@app.route('/acceptedfiles')

def acceptedfiles():

    sql="select SIno,FileOwner,Dataprovider,Filename,Action from
dataowneruploadfiles where user='%s' and Action='accept'"%(session['user'])

    data=pd.read_sql_query(sql,mydb)

    return
render_template('acceptedfiles.html',cols=data.columns.values,rows=data.values.
t olist())

@app.route('/getkey/<a>/<b>')

```

```

def getkey(a=0,b=""):

    print(a,b)

    session['id'] = a

    session['filename'] = b

    sql="select Fileencdata from dataowneruploadfiles where Slno='%s' and
Filename='%s'"%(session['id'],session['filename'])

    cur.execute(sql)

    data=cur.fetchall()

    mydb.commit()

    encdata=data[0][0]

    session['encdata']=encdata


    return render_template('getkey.html',encdata=encdata)

```

53

```

@app.route('/securekey',methods=["POST","GET"])

def securekey():

    if request.method=="POST":

        key1=request.form['key1']

        key2=request.form['key2']

        sql="select AES_DECRYPT(Fileencdata, 'key') from dataowneruploadfiles
where Hash1='%s' and Hash2='%s' and Slno='%s' and
Filename='%s'"%(key1,key2,session['id'],session['filename'])

        # sql="select * from dataowneruploadfiles where Hash1='%s' and Hash2='%s'
and Slno='%s' and Filename='%s'"%(key1,key2,session['id'],session['filename'])

        cur.execute(sql)

        data=cur.fetchall()

        mydb.commit()

        data=[j for i in data for j in i]

```

```

    if data==[]:

        msg="You Have entered invalid keys"

        return render_template('getkey.html',
encdata=session['encdata'],msg=msg)

        content=data[0].decode()

        return render_template('final.html',data=content)

    return

render_template('getkey.html',encdata=session['encdata'])

@app.route('/userlogout')

def userlogout():

    session.pop('user',None)

    return redirect(url_for('index'))

```

54

## B. SCREENSHOTS

Home Page:

Data Owner Login:

Data Owner Registration Page:

Owner Home Page:

Upload Page:

View Files Page:

Send Request:

Data owner secured files:

57

File Content:

Manager Login Page:

Manager Home Page:

DD Providers page:

58

Data Providers list:



Providers Login Page:

59

Data providers Home Page:

Data Owners Request:

View Secured Files:

User Requests:

Data User Login Page:

Data User Registration:

61

Data User home page:

Data User Files:

62

Data requested Files:

Data Decrypt Page:

Data User Decrypted Page:

**TEST CASES:**

Input	Output	Result
Input text files	File upload or not	Success

**Test cases Model building:**

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read data	File data.	Data read successfully	Data read success.	P
2	Performing Encryption on file data	Encryption has to perform on file data	Encryption should be performed on file data	Encryption successfully completed.	P

3	Generating Key pair	Key has to generate	Key will generate	Key Generated successfully.	P
4	Cipher text	File data Encrypted data will Decrypt	Data should be decrypt	Data decrypted successfully	P

## C. Research paper :- Data Storage and Sharing Solution

§

**Abstract** – An enterprise that requires essential digital data might select from storage and sharing options that have been approved by an established Internet provider. However, there are several issues to consider: (1) how can the organization's identity be protected on certificates? (2) How can sensitive digital data stored on the system be protected? (3) How can data sharing be made secure, open, and equitable? In what ways can the validity of shared data be verified without compromising privacy? To this end, we offer a variety of formats for creating, storing, and sharing data. We implement a group signature system for a collection of trustworthy businesses that offer the same kind of service in the data-producing schema, and each business in the collection produces valuable digital data from the raw data received from the others. Data owners upload their data to the Inter-Planetary File System network and store the corresponding access address and certificate on the blockchain ledger using the data storage schema. In accordance with the data sharing schema, everyone could verify the reliability of shared data before requesting data sharing with the data owner. Smart contracts are used to share data, and to ensure honesty, so that the involved parties should escrow. Among the security properties guaranteed by schemas for storing and sharing data are confidentiality, anonymity, integrity, and privacy.

**Key Words:** Distributed Ledger Technology (DLT), Meaningful Data (MD), Reputable Organization (RO), data owner (DO) and Decentralized Storage.

### 1. INTRODUCTION

The amount of data has increased worldwide at an exponential rate, and reliable data are among the most significant resources for both consumers and businesses. By 2025, there is no doubt that the amount of data generated will increase and stored globally will reach 175 zettabytes. Also, it is predicted that 5 billion people will be using data

every day by the year 2025 [1]. Due to this, there is a high demand for important data storage and sharing, which creates problems with data security. There are now two basic architectures, centralised and decentralised systems, which are utilised for data storing and sharing.

Businesses can keep data on their datacentre system for the centralised architecture. The running expenses and scalability of these systems are substantial, albeit [2]. Cloud storage services are better suitable for Internet

of Things (IoT) systems and can save costs and allow for flexible system growth. Studies in [3]–[7] have looked into the pairing of IoT and cloud storage services. Data sharing and storing must be done in a secure and private manner access control models and encryption algorithms have been developed in [8] through [11]. SECURED L was proposed by Murat Kantarcioglu et al. [12] for the protection of databases. Hold sensitive information. However, [13] mentions two drawbacks of the centralised architecture: (1) Data security: By system administrators or intruders who have hacked the system, stored data may be viewed, altered, or removed without authorization; (2) Availability: When the centralised systems breakdown as a result of system overload, DoS or Users cannot access the services due to DDoS assaults or system issues. Due to its characteristics like anonymity, transparency, decentralisation, and auditability, blockchain (BC) technology is typically used as the key component in decentralised architectural solutions [14]–[16]. The shared data currently impossible to verify BC network data accuracy and dependability using current technologies. Particularly, information that has been validated and certified by a reputable organisation (RO) is regarded as significant information (MD). For instance, MD, a respectable medical institution with highly qualified doctors, publishes a from a digital medical record's diagnostic findings. In addition to being completely shared or commercialised with other users or organisations on the network, the data owner (DO) must keep

MD safely stored on the system (DO). When agreeing to carry out data sharing techniques must guarantee that requesters may confirm the legitimacy and accuracy of the data provided. The data shared through the BC network's accuracy and dependability. Data regarded to be significant are those that have been verified and validated by an established organisation (RO) (MD). For instance, in the realm of medicine, MD, A diagnostic outcome from A well-known medical organisation with highly skilled physicians has

released a digital health record. The MD of a data owner (DO) must be stored securely on the system before it may be freely distributed or financially exploited by other network users or organisations. When opting to carry out a data-sharing contract, data sharing procedures must make sure that requesters can validate the validity and correctness of shared data.

The traditional data sharing approach relies on trust between the two participants taking part

66

in the exchange process for the integrity of shared data to be maintained. For instance, hospitals and doctors firmly believe that the medical records they receive from patients are accurate. In certain circumstances, RO must provide anonymity in MD created by them. Also, it is important to protect DOs' privacy because nobody should be aware of which RO's service they employed. The identity of persons involved in the sharing process must also be kept anonymous, and the It is necessary to confirm the reliability of the supplied data while protecting its confidentiality. Storage of data is necessary. For approved digital data storage and sharing, systems for data sharing and storing must meet the following criteria: **To store data:** It is necessary to safeguard the security and integrity of the system's data storage as well as the privacy of DO and the anonymity of certificate authorities. **For data sharing:** Before making a request to DO, everyone using the system can confirm the accuracy of supplied data. It's important to remember that nobody can read the supplied data; they can only verify its accuracy. Without using a middleman, data sharing takes place directly between DO and DU. Scalability, availability, and integrity must all be ensured by the technology used for data storage and sharing. However, none of the proposed methods satisfies the aforementioned criteria. In this study, we offer recommendations for data gathering, archiving, and sharing methods. In our minds, RO is a data provider (DP), and DPs that provide similar services join forces to create groups in the data-generating system. A collection is under the control of a group manager. There are many DPs giving the same service. The team's sole DP transforms unprocessed DO data into MD. After that, DP employs a symmetric technique and a private key to encrypt MD. The MD cypher text is used later on by DP to generate a certificate (denoted by EMD). Last but not least, a secure channel will be used to communicate DP's data, the certificate, and EMD to DODO stores EMD in the A block chain network transaction stores the Inter-Planetary File System (IPFS), together with the supporting information and EMD's IPFS access address. However none of the aforementioned criteria are fully satisfied by the options available today. In this research, the researchers propose systems for producing, storing, and exchanging data. When we talk about RO, we refer to it as a data provider (DP), and DPs that provide related services join forces to form groups. A group manager assembles DPs that

provide a similar service in the system that produces data. The

group's DO raw data is transformed into MD using a specific DP. Then MD is encrypted by DP using a symmetric method and a private key. Afterward, using the MD cipher text, DP generates a certificate (denoted by EMD). Lastly, a secure channel will be used to communicate with DO and send EMD, the certificate, and DP's data. Data is stored via an Inter-Planetary File System by DO (IPFS); a transaction on the blockchain system stores EMD's IPFS access address as well as relevant data.

## 2. METHODOLOGY AND

### ALGORITHMS 2.1 Modules

#### 2.1.1. Data Owner

Login: Data owner must sign in using the same details given upon registration.

Register: Each and every Data owner has to register.

Upload files: uploads his files into the cloud. View Files: views all the files which are uploaded by the him/her and send request if there is any requests.

Secured Files: He can view his/her files are secured or not.

Logout: Finally logout from the system.

#### 2.1.2. Data Provider

Login: data provider has to login with valid details which are used in his / her Registration

Register: Each and every data provider has to register and management has to accept request.

View Data Owners Request: Views data owner's requests and secure those files. View Secured Files: data provider can view all files which are secured.

User Requests: views user requests for files and sends keys to the users.

Logout: Finally logout from the system.

#### 2.1.3. Group Manager

Login: Manager will login with default details

Add providers: He adds the data providers and views all the data providers who are added by him

Logout: Finally logout from the system.



Login: Data user will login with his credentials which are added in his/her registration. View Files: He views all the files which are uploaded by the data provider.  
 Send Request: sends the request for file My Files: views all the files which are accepted Check Files: he need hash codes to view the file content  
 Download file: if hash codes are valid the data will be decrypted and can download as txt file  
 Logout: Finally logout from the system.

## 2.2 Cloud

Three fundamental services comprise the cloud:

**2.2.1 Software-as-a-service (SaaS)** involves giving customers is given software licences. Licenses are typically made accessible on demand or under a pay-as-you-go system. The Microsoft Office 365 suite includes a similar technique.

**2.2.2 Infrastructure-as-a-service (IaaS)** refers refers to a process for delivering any resource, such as As part of an on-demand service utilizing IP-based communication, Storage, servers, and operating systems are offered. Customers can access these resources through an on-demand, outsourced service rather than acquiring their own servers or software. Two popular IaaS implementations are both Microsoft Azure and IBM Cloud.

**2.2.3 Platform-as-a-service (PaaS)** is among it is regarded as the most complex of the three cloud-based computing layers. While SaaS and PaaS are similar in some respects, As opposed to SaaS, PaaS is essentially a platform for the development of online-available software. The opposite is true here. This technique includes using platforms like Heroku and Salesforce.com.

## 2.3 Data Encryption

Information is changed or encoded using data encryption so that only those with the proper password or secret code (technically referred to as a decryption key) can decode it. The term "plaintext" refers to unencrypted data, whereas "cypher text" is used to describe encrypted data. Encryption is currently one of the most extensively used and effective data security methods used by businesses. Asymmetric encryption, The two primary techniques for data encryption are symmetric encryption and often-referred-to public-key

encryption.

## 2.4 Purpose

As digital information is stored on electronic devices and transferred across computer networks, data encryption is used to protect the privacy of that information. Modern encryption algorithms have taken the place of the antiquated Data Encryption Standard (DES), which are crucial for ensuring the safety of IT systems and communications.

**Fig -1: Encryption and Decryption of data**

These algorithms support confidentiality and power important security projects like authentication, integrity, and non-repudiation. Authentication and proving that a message's either the message's contents remain unchanged from when it was transmitted, or integrity and establish its origin. A message's sender cannot change their mind about their conduct thanks to non-repudiation.

## 2.5 Data Decryption

After data has been encrypted and rendered unreadable, it can be restored to its original state through the process of decryption. The system takes the jumbled data and extracts and turns it into texts and pictures that both the decryption system and the reader can understand with ease. Both manually and automatically can carry out the decryption operation. It might also be accomplished using a set of keys or a password. An encryption decryption system should be in place for several reasons, one of which is privacy. Unauthorized people or organizations can read and access information that is transferred across the World Wide Web. As a result, data is encrypted to prevent theft and loss. A handful of the things that are typically encrypted include emails, text documents, images, user information, and directories. Before granting access to encrypted data, the person in charge of decryption sees a window or prompt that asks for a password.

## 2.6 Block Chain

Data can be stored using the blockchain in a way

that makes fraud, system modifications, and hacking challenging or impossible. A block chain is a network of computer systems that copies and disseminates a digital record of transactions across the whole network, according to the most basic description. Every new transaction that takes place is copied to each participant's ledger, and each block in the block chain comprises numerous transactions. The decentralised database that is controlled by many users is referred to as "distributed ledger technology" (DLT). A hash, an immutable cryptographic signature, serves as a permanent record of transactions on a blockchain. This suggests that it would be obvious that a block in a chain had been altered if one of the blocks in the chain was changed. Hackers would need to change every block in every distributed version of the chain if they wanted to compromise a block chain system.

## 2.7 SHA Algorithm

The term "secure hashing algorithm" is utilized. A modified version of the MD5 method called SHA is used to hash data and certificates. A hashing algorithm uses bitwise operations, modular additions, and compression techniques to reduce the input data into a more digestible, unintelligible format. You might be asking whether hashing can be broken or decrypted. Hashing is one-way, which means that after when data is hashed, the digest that is produced can only be decoded by a brute force attack. This is the only way that hashing differs from encryption. The diagram below illustrates how the SHA algorithm functions. Because of how SHA operates, if even one character in the message changes, a different hash will be produced. As shown in Fig. 2, two messages that are similar yet different— they discussed Heaven and Heaven Are Not the Same. Yet, the sole distinction between a capital and small letter is that.

**Fig -2:** Hashing of two similar, but different messages

## 2.8 Implementation

### Fig -3: Architecture of Data Storing and Sharing

Data owner is the person who login's into the system after registration and upload his/her files into the cloud and view that file is secured or not .Data owner can view his files content using hash codes and if there is any improper content in that file, he can delete the complete file and upload the file with proper content. Data provider is the person who gives the protection to the data owner's uploaded files by using encryption technique and view his/her secured files and view user requests for files if is there any requests from users he can send keys to the user for decryption. Data providers are added by the Group manager. Group manager plays a vital role like adding data providers and views the data providers and finally logout from the system. Data user is the person who can view all files which are uploaded by the data owner and sends request to the data provider if he accepts then the key will be generated for user and sent to the user for decrypting the encrypted data. These operations are performed by the data user after successful login and finally logout from the system.

## 3. CONCLUSIONS

The three tactics recommended in this study are gathering data, storing it, and distributing it. Under the data, a group manager groups DPs that provide the same kind of service. Production plan. From RD sent by DO, DP can generate MD, and then issue a certificate on

EMD. Our system provides DP anonymity, DO privacy, and integrity of the stored data along with security Prior to sending data to DO for distribution, users can independently verify its accuracy using this approach. It should be noted that everyone has access to the shared data in order to verify its reliability but cannot view its contents. Existing solutions were not able to satisfy this property. Moreover, DO and DU exchange data directly, without the use of any middlemen. The proposed systems satisfy the requirements for A assessment of security aspects revealed non-repudiation, anonymity, confidentiality, integrity, and privacy.

## REFERENCES

- [1] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world from edge to core," IDC White Paper, Nov. 2018.
- [2] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *EEE Commun. Surveys Tuts*, vol. 15, no. 2, pp. 909–928, 2nd Quart. 2013.
- [3] L. Jiang, L. D. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An IoT-oriented data storage framework in cloud computing platform," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1443–1451, May 2014.
- [4] T. A. Phan, J. K. Nurminen, and M. Di Francesco, "Cloud databases for Internet- of-Things data," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom)*, Sep. 2014, pp. 117–124.
- [5] K. Yasumoto, H. Yamaguchi, and H. Shigeno, "Survey of real-time processing technologies of IoT data streams," *J. Inf. Process.*, vol. 24, no. 2, pp. 195–202, 2016.
- [6] A. Kumar, N. C. Narendra, and U. Bellur, "Uploading and replicating Internet of Things (IoT) data on distributed cloud storage," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2016, pp. 670–677.
- [7] K. Hossain, M. Rahman, and S. Roy, "IoT data compression and optimization techniques in cloud storage: Current prospects and future directions," *Int. J. Cloud Appl. Comput.*, vol. 9, no. 2, pp. 43–59, Apr. 2019.
- [8] J. D. Bokefode, A. S. Bhise, P. A. Satarkar, and D. G. Modani, "Developing a secure cloud storage system for storing IoT data by applying role based encryption," *Procedia Comput. Sci.*, vol. 89, no. 1, pp. 43–50, 2016.
- [9] W. Wang, P. Xu, and L. T. Yang, "Secure data collection, storage and access in cloud- assisted IoT," *IEEE Cloud Comput.*, vol. 5, no. 4, pp. 77–88, Jul. 2018.
- [10] M. Rashid, S. A. Parah, A. R. Wani, and S. K. Gupta, "Securing Ehealth IoT data on cloud systems using novel extended role based access control model," in *Internet Things (IoT)*. Cham, Switzerland: Springer, 2020, pp. 473–489.
- [11] R. Arora, A. Parashar, and C. C. I. Transforming, "Secure user data in cloud computing using encryption algorithms," *Int. J. Eng. Res. Appl.*, vol. 3, no. 4, pp. 1922– 1926, 2013.
- [12] M. Kantarcioglu and F. Shaon, "Securing big data in the age of AI," in *Proc. 1st IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl. (TPSISA)*, Dec. 2019, pp. 218–220.
- [13] C. Zhang, J. Sun, X. Zhu, and Y. Fang, "Privacy and security for online social networks: Challenges and opportunities," *IEEE Netw.*, vol. 24, no. 4, pp. 13–18, Jul./Aug. 2010.
- [14] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. Sebastopol, CA, USA: O'Reilly Media, 2014.
- [15] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.