1. INTRODUCTION

Cloud computing is a distributed computing model based on a large shared virtualized computing resource pool, it helps users use powerful computing and storage resources. And it can greatly reduce the burden of data storage on hardware and software for users, which encourages many enterprises and individuals to store their data on cloud servers.

Despite the great success of cloud storage, it also faces various challenges, and its security, reliability and privacy have always been a serious issue. After the user stores the data on the cloud server, the server provider may damage or delete the user data due to various factors, verifying the integrity of outsourced data becomes a crucial issue in cloud storage. Remote data integrity audit technology is very convenient and safe to help users check the integrity of data stored in outsourced. Therefore, the essence of cloud data security is how cloud storage providers (CSP) can establish trust with users. Cloud device failures, illegal attacks, and CSPs may be bribed to view user data, all of which can lead to illegal infringement of user data. Furthermore, even if the user data is damaged, the user may not be able to hold the CSP accountable effectively, since the CSP may evade responsibility and deny it. This is due to the lack of trust between the two parties, resulting in the party being questioned being unable to come up with evidence that would convince the other party. In addition, the current law on cyber security is not sound, which makes it difficult for users to obtain due compensation.

In traditional cloud auditing schemes, there is an entity called auditors (often referred to as third-party auditors, or TPA) which implement public audits. The TPA accept audit mandates from data owners and perform as instructed. In each of these methods, a trusted Third Party Auditor (TPA) must be found to assist the user in auditing, but in reality it is difficult to find fully trusted third-party auditors. For example, TPA will also partner with CSP for some ulterior purpose to hide data corruption, or with data owners to avoid penalties.

The emergence of block chain can solve this problem very well. Block chain has the properties of decentralization, tamper resistance, consistency and traceability. Therefore, information stored on the block chain is open and transparent. In recent years, more and more researchers use block chain to replace third-party auditors. Although the use of block chain as a trusted third-party auditor can well address users' concerns in cloud computing environments, but the rapid growth of blocks will lead to high cost for block chain network maintenance and for user creation of new blocks.

To solve the above problems, a data integrity verification scheme based on block chain expansion technology is proposed. By slowing the growth of the block chain, reducing storage and calculation costs. In particular, our contribution can be summarized in three aspects:

1) A data integrity audit protocol based on plasma smart contracts is proposed. By introducing plasma sub-chains and deploying smart contracts on the main chain and sub-chains, the storage pressure of the main chain can be reduced and the growth rate can be slowed down through this protocol. TPA audit protocol can be executed with low computational and communication overhead.

2) A batch auditing scheme is proposed, the scheme can batch-process multiple audit tasks at the same time. In order to avoid affecting the user experience due to the communication with the CSP during the audit process as much as possible, the concept of non-interactive audit is introduced. For the sake of ensuring the correctness of the audit, the reward pool mechanism is adopted, and the verification node can obtain reasonable rewards.

3) An analysis of the security of the scheme shows that it can achieve the expected security objectives. Numerous experiments on the ether block chain also showed the efficiency and effectiveness of the scheme.

1.1 Motivation

1. The Identify Motivation Data: Determine the types of motivation data that need to be recorded and audited. This could include performance metrics, rewards, incentives, feedback, etc.

2. Blockchain Selection: Choose a suitable blockchain platform or framework. Consider factors like scalability, consensus mechanism, security features, and smart contract

capabilities. Ethereum, Hyperledger Fabric, or a specialized blockchain for audit trails might be options.

3. Smart Contract Development: Develop smart contracts to manage the motivation data transactions. Smart contracts can automate processes like recording motivation data, verifying its integrity, and executing predefined actions based on specified conditions.
4. Data Encryption and Hashing: Encrypt sensitive motivation data before storing it on the blockchain. Use cryptographic hashing algorithms to generate unique fingerprints (hashes) for each piece of data. This ensures data integrity and prevents tampering.
5. Permissioned Access: Implement access control mechanisms to restrict who can read, write, or modify motivation data on the blockchain. This prevents unauthorized tampering and ensures that only authorized parties can access relevant information.
6. Timestamping: Leverage the blockchain's inherent timestamping capabilities to record the time when motivation data is added or updated. This creates an immutable record of events, facilitating audit trails and ensuring transparency.

1.2 Problem Definition

The primary challenge is to ensure the integrity and authenticity of data, especially in scenarios where multiple parties are involved, and there is a need for trust without reliance on a central authority. Traditional databases or centralized systems may be susceptible to data tampering, unauthorized access, or single points of failure. The goal is to design a solution that provides a tamper-proof, transparent, and decentralized mechanism for auditing data integrity.

1.3Objective

The primary goal is to ensure the integrity, authenticity, and transparency of data stored and transacted within a system or network. The scheme aims to provide a tamper-proof and immutable record of data transactions, enabling trustworthy audit trails without reliance on centralized authorities. Blockchain serves as a decentralized, distributed ledger that records transactions in a transparent and immutable manner. Utilizing blockchain technology ensures data integrity by cryptographically linking and timestamping each transaction across a network of nodes.

- Smart Contracts: Self-executing contracts with predefined rules and conditions. They automate data transactions and ensure compliance with established protocols.
- Consensus Mechanism: Determines how transactions are validated and added to the blockchain, ensuring agreement among network participants without the need for a central authority.
- Cryptographic Hashing: Generates unique fingerprints (hashes) for data, ensuring tamper-proofing and enabling verification of data integrity.
- Permissioned Access: Controls access rights to data on the blockchain, ensuring only authorized parties can read, write, or modify information.
- Timestamping: Records the time of data transactions, creating a chronological sequence of events that serves as an immutable audit trail.

2. LITERATURE SURVEY

Y. Fan et al., [1] of One Secure Data Integrity Verification Scheme for Cloud Storage paper presents a scheme that addresses the challenge of maintaining data integrity in the cloud, where data is stored on remote servers managed by third-party providers. The authors introduce a secure data integrity verification scheme designed to detect any unauthorized modifications or tampering of data stored in the cloud. Key features of the proposed scheme include the utilization of cryptographic techniques such as hash functions and digital signatures to generate and verify data integrity proofs. These proofs are used to verify the integrity of data stored in the cloud without the need for users to retrieve the entire data file. The scheme ensures the confidentiality of user data by encrypting it before transmission to the cloud storage provider. This encryption helps prevent unauthorized access to sensitive information stored in the cloud.

The authors as evaluated the performance and effectiveness of their proposed scheme through theoretical analysis and experimental simulations. The results demonstrate the scheme's capability to efficiently detect data tampering while maintaining low overhead in terms of computational resources and communication overhead. Hence this paper provides a comprehensive and practical solution to the challenge of ensuring data integrity in cloud storage environments. The proposed scheme offers a balance between security, efficiency, and usability, making it suitable for real-world applications in cloud computing.

Q. Zhou et at., [2] of Solutions to Scalability of Blockchain says that, as the Blockchain technology has gained significant attention in recent years due to its potential to revolutionize various industries by providing decentralized and immutable ledger systems. However, one of the primary limitations of blockchain is its scalability, particularly concerning transaction throughput and confirmation latency.

The survey conducted by Zhou et al. aims to provide a comprehensive overview of the scalability issues in blockchain and the diverse range of solutions proposed to mitigate these challenges. The paper categorizes these solutions into several key areas, including consensus mechanisms, sharding techniques, off-chain solutions, and layer-2 scaling solutions. Consensus mechanisms play a crucial role in blockchain scalability, and the survey explores various consensus algorithms such as Proof of Work (PoW), Proof of Stake (PoS), and their variations. Additionally, the paper delves into the concept of sharding, which involves partitioning the blockchain network into smaller shards to improve throughput. Furthermore, the survey discusses off-chain solutions like payment channels and sidechains, which facilitate faster and cheaper transactions by moving some operations off the main blockchain. Layer-2 scaling solutions, including state channels and plasma, are also examined for their potential to enhance scalability without compromising security.

Overall, the survey provides a comprehensive overview of the scalability challenges in blockchain technology and the diverse range of solutions proposed by researchers and developers. By categorizing and analyzing these solutions, the paper offers valuable insights into the ongoing efforts to overcome the scalability limitations of blockchain systems.:

K. Xu et at., [3] of Blockchain-based integrity verification of data migration in multi-cloud storage says that, the Data migration is a critical challenge in multi-cloud storage systems, involving the transfer of large volumes of data between different cloud storage locations. Ensuring the integrity and security of data during migration is paramount to prevent unauthorized access, tampering, or loss of data.

This paper examines various blockchain-based integrity verification techniques and frameworks proposed in the literature. These techniques leverage smart contracts, cryptographic hashing, and distributed consensus mechanisms to create a tamper-proof audit trail of data migration transactions. By recording transaction details, such as data origin, destination, timestamp, and cryptographic hash values, on the blockchain, these solutions enable stakeholders to verify the integrity and authenticity of migrated data. Furthermore, the survey discusses the challenges and limitations of existing blockchain-based integrity verification approaches, such as scalability, performance, and interoperability issues. It also identifies potential areas for future research and development to enhance the effectiveness and efficiency of blockchain-based integrity verification in multi-cloud storage environments.

Overall, the paper provides a comprehensive overview of the current state-of-the-art in blockchain-based integrity verification for data migration in multi-cloud storage systems, offering valuable insights for researchers, practitioners, and stakeholders in the field.

G. Xie et at., [4] says that Cloud computing offers numerous benefits for data storage and processing but raises concerns about data integrity and security. Traditional methods for data integrity verification often lack efficiency and scalability, especially in large-scale cloud environments. To address these challenges, the authors propose a blockchain-based scheme for cloud data integrity verification, leveraging the decentralized and immutable nature of blockchain technology.

This paper provides an in-depth exploration of the key components and mechanisms of their proposed scheme. It describes how blockchain technology can be utilized to create a tamper-proof ledger of data integrity verification transactions, ensuring transparency, immutability, and traceability of data integrity proofs. Also discusses the architecture and design principles of the blockchain-based integrity verification scheme, including the roles of different entities such as data owners, cloud service providers, and verifiers. It also examines the cryptographic techniques and consensus mechanisms employed to secure data integrity proofs and maintain the integrity of the blockchain ledger.

Furthermore, the survey evaluates the performance and efficiency of the proposed scheme compared to traditional integrity verification methods. It analyzes factors such as computation overhead, storage requirements, and verification latency, demonstrating the effectiveness of the blockchain-based approach in achieving high efficiency while ensuring strong security guarantees.

Overall, the paper provides a comprehensive overview of the blockchain-based cloud data integrity verification scheme proposed by Xie et al., highlighting its efficiency and effectiveness in addressing data integrity challenges in cloud computing environments. The survey offers valuable insights for researchers, practitioners, and stakeholders interested in leveraging blockchain technology for secure and efficient data management in the cloud.

3. ANALYSIS

3.1 Existing System

Fan *et al.* zhe'ge replaced the TPA with a smart contract, and the user signed an agreement with the CSP to prevent one party from denying it. The data owner obtains the hash of the remote data through the block identifier and compares it with the hash value previously stored in the blockchain ledger. Obviously, this scheme cannot resist the replay attack carried out by the CSP. Yu *et al.* decentralize the data without any TPA in their scheme. Their solution is effective against replay attacks due to the random challenge set generated in each audit request. To defend against dishonest provers and verifiers, Xu *et al.* proposed an arbitrable data audit protocol that supports exchange hashing. Existing cloud storage service providers (CSP) may not have a fair compensation for users even if they damage data, and CSP may store redundant and duplicate data. Yuan *et al.* proposed a deduplication scheme with public audit and fair arbitration.

Zhou *et al.* proposed a solution for blockchain scalability. The existing expansion schemes are designed to improve different layers, and are divided into layer-0 expansion, on-chain expansion, and off-chain expansion. Among them, on-chain expansion improves the efficiency of the blockchain by changing the basic protocol. Off-chain expansion does not change the basic protocol, and changes are made at the application layer to improve scalability. Layer-0 expansion improves blockchain scalability by changing the underlying data transmission protocol of the blockchain. The on-chain expansion scheme includes data layer improvement scheme, consensus layer improvement scheme and network layer improvement scheme. The basic idea is to increase the block size (either directly or indirectly) or reduce the block verification propagation time and consensus formation time. The off-chain expansion scheme mainly includes four methods: state channel, side chain, cross-chain and off- chain computation. The idea is to transfer some on-chain transactions to off-chain for execution, in order to reduce the processing pressure on the chain and improve the overall efficiency. While improving the performance of the blockchain, the off-chain scaling technology takes into account decentralization and security, and has various excellent properties.

Disadvantages

- The system is not implemented Data auditing technique for data integrity proof.
- The system is not implemented Data Hashing Techniques.

3.2 Proposed System

In the proposed system, A data integrity audit protocol based on plasma smart contracts is proposed. By introducing plasma sub-chains and deploying smart contracts on the main chain and sub-chains, the storage pressure of the main chain can be reduced and the growth rate can be slowed down through this protocol. TPA audit protocol can be executed with low computational and communication overhead. In the proposed system, A batch auditing scheme is proposed, the scheme can batch-process multiple audit tasks at the same time. In order to avoid affecting the user experience due to the communication with the CSP during the audit process as much as possible, the concept of non-interactive audit is introduced. For the sake of ensuring the correctness of the audit, the reward pool mechanism is adopted, and the verification node can obtain reasonable rewards. In the proposed system, An analysis of the security of the scheme shows that it can achieve the expected security objectives. Numerous experiments on the ether block chain also showed the efficiency and effectiveness of the scheme.

Advantages

- Batch auditing: including multi-user single-task auditing and multi-user multi-task auditing. This is to ensure the efficiency of auditing.
- Public auditing: Ensure that any user including the data owner can challenge the CSP to verify the integrity of the data based on the certificate generated by the CSP.

3.3 System Requirement Specification

Software Requirements

1)	Operating System	-	Windows XP
2)	Coding Language	-	Java/J2EE(JSP,Servlet)

3)	Front End	-	J2EE
4)	Back End	-	MySQL

Hardware Requirements

1)	Processor	-	Pentium –IV
2)	RAM	-	4 GB (min)
3)	Hard Disk	-	20 GB
4)	Key Board	-	Standard Windows Keyboard
5)	Mouse	-	Two or Three Button Mouse
6)	Monitor	-	SVGA

3.3.1 Purpose

The purpose of a data integrity audit scheme based on blockchain technology is multifaceted and serves several key objectives:

1. Ensuring Trustworthiness: The primary purpose is to ensure the trustworthiness of data stored and transacted within a system or network. By leveraging blockchain's decentralized and immutable ledger, the scheme aims to prevent unauthorized alterations or tampering of data, thereby maintaining its integrity.

2. Transparency and Accountability: Another purpose is to promote transparency and accountability in data transactions. Blockchain technology provides a transparent record of all transactions, visible to authorized parties, enabling stakeholders to verify the authenticity and integrity of data without relying on intermediaries.

3. Mitigating Risks of Fraud and Manipulation: By providing a tamper-proof audit trail, the scheme helps mitigate the risks of fraud, manipulation, and unauthorized access to data. Cryptographic hashing algorithms ensure that any changes to data are detectable, enhancing security and reducing the likelihood of malicious activities.

4. Streamlining Auditing and Compliance Processes: The scheme aims to streamline auditing and compliance processes by automating data transactions and

ensuring adherence to predefined rules and regulations. Smart contracts enforce compliance with established protocols, reducing the need for manual intervention and enhancing efficiency.

5. Facilitating Regulatory Compliance: Compliance with legal and regulatory requirements is a crucial purpose of the scheme. By adhering to data protection regulations (e.g., GDPR) and industry standards, organizations can ensure that their data handling practices meet the necessary legal obligations, thereby reducing the risk of non-compliance penalties.

6. Enhancing Data Security and Privacy: Protecting the security and privacy of data is fundamental to the scheme's purpose. Encryption techniques and access controls ensure that sensitive information remains confidential, while cryptographic hashing and consensus mechanisms safeguard data integrity against unauthorized modifications.

7. Enabling Traceability and Auditability: The scheme enables traceability and auditability of data transactions, allowing stakeholders to track the history of data.

3.3.2 Scope

The scope of a data integrity audit scheme based on blockchain technology encompasses various aspects related to ensuring the integrity, security, and transparency of data transactions. Here's an outline of the scope:

1. Data Types: The scheme covers a wide range of data types, including but not limited to financial transactions, supply chain records, healthcare data, legal documents, academic credentials, and IoT sensor data. It aims to ensure the integrity of both structured and unstructured data stored and transacted within the system.

2. Data Lifecycle: The scope includes the entire data lifecycle, from creation and transmission to storage and disposal. It encompasses processes such as data entry, validation, update, retrieval, and archival, ensuring that data integrity is maintained throughout its lifecycle.

3. Blockchain Platforms: The scheme may consider different blockchain platforms or frameworks suitable for the intended application, such as public blockchains (e.g.,

Ethereum, Bitcoin), permissioned blockchains (e.g., Hyperledger Fabric, Corda), or specialized blockchain solutions tailored to specific use cases.

4. Smart Contracts and Automation: It involves the use of smart contracts to automate data transactions and enforce predefined rules and conditions. Smart contracts facilitate self-executing agreements, ensuring transparency, efficiency, and accuracy in data management processes.

3.3.3 Overall Description

An overall description of a data integrity audit scheme based on blockchain technology involves the integration of blockchain principles to ensure the integrity, security, and transparency of data transactions. Here's an overview:

1. Objective: The primary objective is to establish a tamper-proof and transparent system for managing data transactions, ensuring that data integrity is maintained throughout its lifecycle.
2. Technology Stack: The scheme leverages blockchain technology, including distributed ledger technology, cryptographic hashing, smart contracts, and consensus mechanisms, to achieve its objectives.
3. Data Types: The scheme accommodates various types of data, ranging from financial transactions and supply chain records to healthcare data and IoT sensor readings.
4. Blockchain Platform: Depending on the specific requirements and use case, the scheme may utilize a suitable blockchain platform such as Ethereum, Hyperledger Fabric, or a specialized blockchain solution tailored to the application domain.
5. Smart Contracts: Smart contracts are employed to automate data transactions and enforce predefined rules and conditions. These self-executing contracts ensure transparency, accuracy, and efficiency in data management processes.

4. DESIGN

4.1 UML DIAGRAMS

UML (Unified Modelling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. 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 comprises two major components: a Meta-model and a notation The Unified Modelling Language is used 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 object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Various UML Diagrams are:

- ❖ Use Case Diagram
- ❖ Class Diagram
- ❖ Activity Diagram
- ❖ Sequence Diagram
- ❖ State Chart Diagram
- ❖ Collaboration Diagram
- ❖ Component Diagram
- ❖ Deployment Diagram

4.1.1 USE CASE DIAGRAM

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural 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.

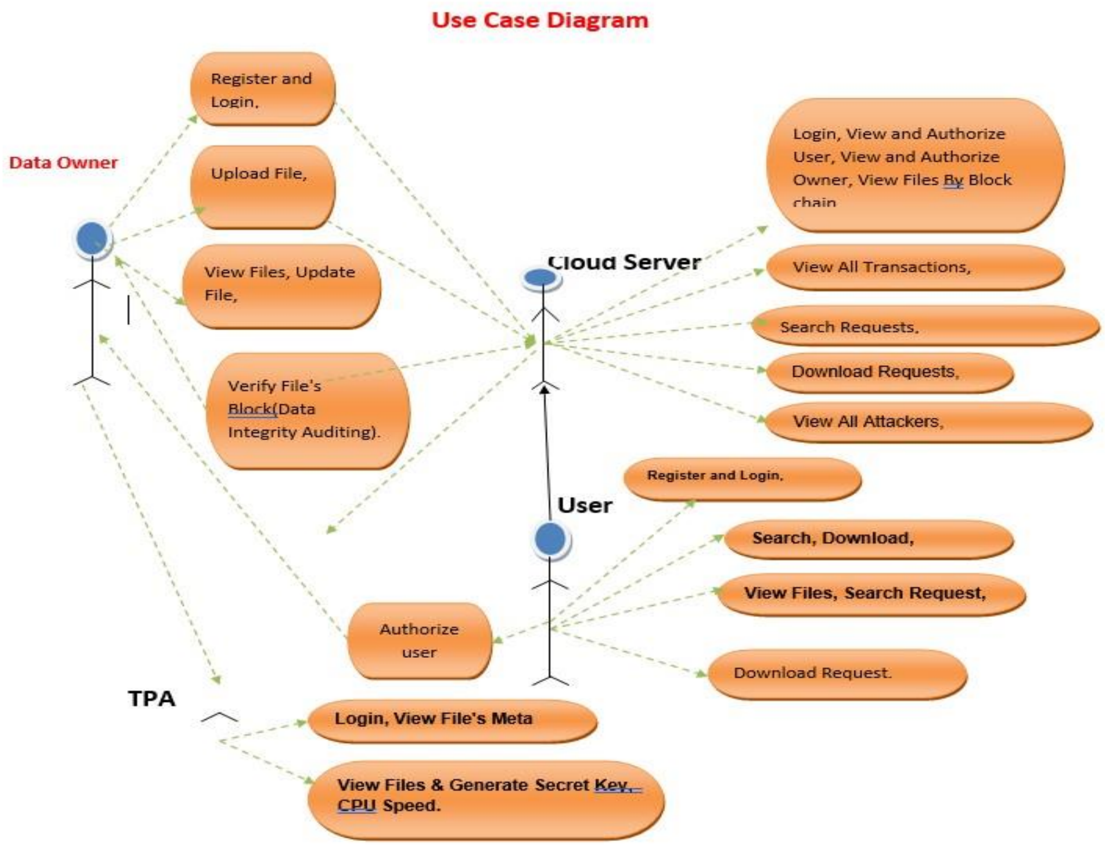


Fig.4.1.1. Use Case Diagram

4.1.2 Class Diagram

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

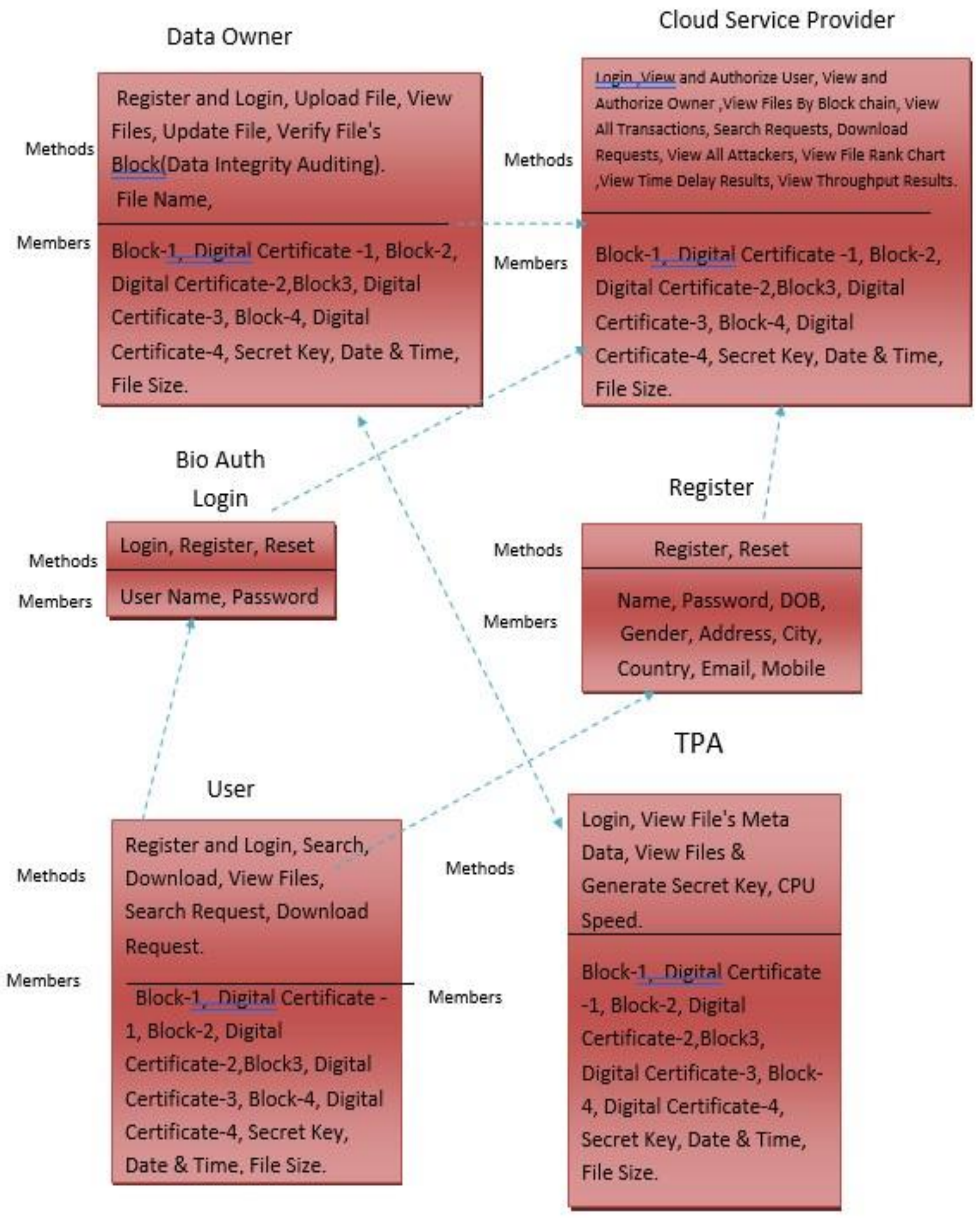

Data Owner
Methods
Register and Login, Upload File, View Files, Update File, Verify File's Block(Data Integrity Auditing). File Name,
Members
Block-1, Digital Certificate -1, Block-2, Digital Certificate-2,Block3, Digital Certificate-3, Block-4, Digital Certificate-4, Secret Key, Date & Time, File Size.
Cloud Service Provider
Methods
Login ,View and Authorize User, View and Authorize Owner ,View Files By Block chain, View All Transactions, Search Requests, Download Requests, View All Attackers, View File Rank Chart ,View Time Delay Results, View Throughput Results.
Members
Block-1, Digital Certificate -1, Block-2, Digital Certificate-2,Block3, Digital Certificate-3, Block-4, Digital Certificate-4, Secret Key, Date & Time, File Size.
Bio Auth Login
Methods
Login, Register, Reset
Members
User Name, Password
Register
Methods
Register, Reset
Members
Name, Password, DOB, Gender, Address, City, Country, Email, Mobile
User
Methods
Register and Login, Search, Download, View Files, Search Request, Download Request.
Members
Block-1, Digital Certificate -1, Block-2, Digital Certificate-2,Block3, Digital Certificate-3, Block-4, Digital Certificate-4, Secret Key, Date & Time, File Size.
TPA
Methods
Login, View File's Meta Data, View Files & Generate Secret Key, CPU Speed.
Members
Block-1, Digital Certificate -1, Block-2, Digital Certificate-2,Block3, Digital Certificate-3, Block-4, Digital Certificate-4, Secret Key, Date & Time, File Size.

Fig.4.1.2. Class Diagram

4.1.3 SEQUENCE DIAGRAM

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

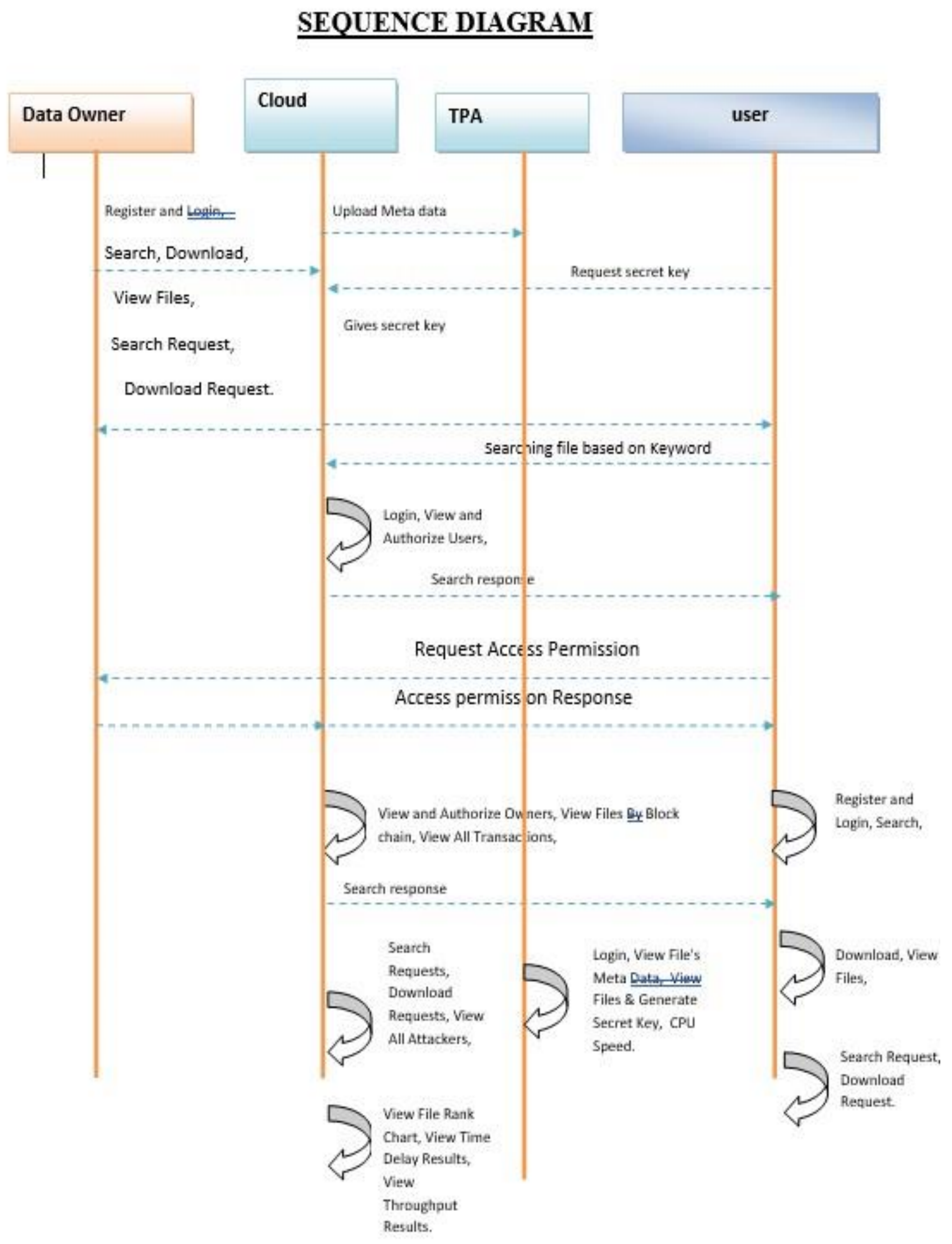

SEQUENCE DIAGRAM
Data Owner
Cloud
TPA
user
Register and ~~Login,~~
Search, Download,
View Files,
Search Request,
Download Request.
Upload Meta data
Request secret key
Gives secret key
Searching file based on Keyword
Login, View and Authorize Users,
Search response
Request Access Permission
Access permission Response
View and Authorize Owners, View Files By Block chain, View All Transactions,
Register and Login, Search,
Search response
Search Requests, Download Requests, View All Attackers,
Login, View File's Meta ~~Data, View~~ Files & Generate Secret Key, CPU Speed.
Download, View Files,
Search Request, Download Request.
View File Rank Chart, View Time Delay Results, View Throughput Results.

Fig. 4.1.3 Sequence Diagram

4.2 SYSTEM ARCHITECTURE

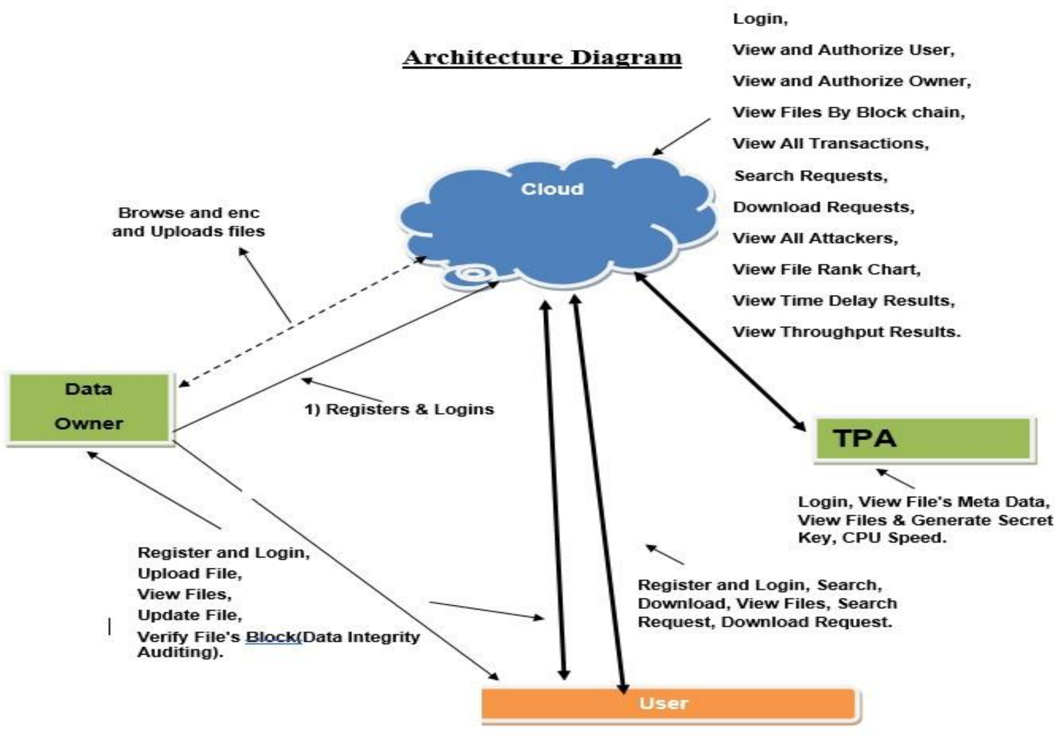

Architecture Diagram
Login,
View and Authorize User,
View and Authorize Owner,
View Files By Block chain,
View All Transactions,
Search Requests,
Download Requests,
View All Attackers,
View File Rank Chart,
View Time Delay Results,
View Throughput Results.
Cloud
Browse and enc and Uploads files
Data Owner
1) Registers & Logins
TPA
Login, View File's Meta Data, View Files & Generate Secret Key, CPU Speed.
Register and Login, Upload File, View Files, Update File, Verify File's Block(Data Integrity Auditing).
Register and Login, Search, Download, View Files, Search Request, Download Request.
User

Fig.4.2. System Architecture

5. IMPLEMENTATION

Implementing a data integrity audit scheme based on blockchain technology involves several key steps and considerations. Here's an overview of the implementation process:

1. Requirement Analysis: Identify the specific requirements and objectives of the data integrity audit scheme. Determine the types of data to be audited, compliance requirements, security considerations, and stakeholder needs.

2. Blockchain Platform Selection: Choose a suitable blockchain platform or framework based on the requirements identified. Consider factors such as scalability, consensus mechanism, privacy features, smart contract capabilities, and interoperability.

3. Smart Contract Development: Develop smart contracts to automate data transactions and enforce predefined rules and conditions. Define the data structures, functions, and logic required to manage data integrity and audit trails on the blockchain.

4. Data Encryption and Hashing: Implement encryption techniques to protect sensitive data before storing it on the blockchain. Use cryptographic hashing algorithms to generate unique fingerprints (hashes) for data, ensuring tamper-proofing and verification.

5.1 Modules

Supervised machine learning: It is one of the ways of machine learning where the model is trained by input data and expected output data. To create such model, it is Necessary to go through the following phases:

1. Data Owner
2. Cloud
3. User
4. TPA

5.2 Module Description

1. **Data Owner:** In this module, data owners securely upload encrypted data to the cloud server, ensuring data confidentiality. Before storage, the data owner encrypts the file to enhance security measures. The data owner retains the capability to manage the encrypted data and execute various operations such as registration, login, file uploading, viewing, updating, and verifying file integrity through block-level auditing. This comprehensive suite of functionalities empowers data owners to maintain control over their data while safeguarding it against unauthorized access or tampering.
2. **Cloud:** The Cloud manages which is to provide data storage service for the Data Owners. Data owners encrypt their data files and store them in the Server for sharing with data consumers. To access the shared data files, data consumers download encrypted data files of their interest from the Server and then Server will decrypt them. The server will generate the aggregate key if the end user requests for file authorization to access and performs the following operations such as Login, View and Authorize User, View and Authorize Owner, View Files By Block chain, View All Transactions, Search Requests, Download Requests, View All Attackers, View File Rank Chart, View Time Delay Results, View Throughput Results.
3. **User:** In this module, users can only access data files using a secret key for enhanced security. They can conduct keyword searches within files, prompting indexing of relevant data on the cloud server. Upon locating matches, the system responds to the user's query. Additionally, users can perform various operations including registration, login, searching, downloading, viewing files, search request and download request. The module facilitates seamless interaction between users and their data, prioritizing security and efficient data retrieval.
4. **TPA:** The TPA module is responsible for certain administrative duties and computational tasks related to file management. Its responsibilities encompass login procedures, viewing file metadata, accessing files, and generating secret keys for secure file access. Leveraging CPU speed, the TPA conducts computational tasks essential for efficient file management and auditing processes. This module facilitates streamlined access to metadata, ensures secure file access through key generation, and optimizes computational resources for effective management and auditing of files stored within the cloud serer.

5.3 Introduction to Technologies Used

1.Blockchain Technology: Blockchain is a decentralized, distributed ledger technology that enables secure and transparent recording of transactions across a network of computers. It utilizes cryptographic techniques to ensure data integrity, immutability, and transparency without the need for a central authority.

2.Smart Contracts: Smart contracts are self-executing contracts with predefined rules and conditions written in code. They automate the execution of transactions and enforce agreements on the blockchain, eliminating the need for intermediaries.

3.Cryptographic Hashing: Cryptographic hashing algorithms generate unique fingerprints (hashes) for data. These hashes are used to verify the integrity of data by detecting any changes or tampering.

4.Consensus Mechanisms: Consensus mechanisms determine how transactions are validated and added to the blockchain. Examples include Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT).

5.Encryption Techniques: Encryption techniques are used to protect sensitive data from unauthorized access. Public-key cryptography and symmetric-key cryptography are commonly employed to secure data transmission and storage.

6.Access Control Mechanisms: Access control mechanisms manage permissions and restrict who can read, write, or modify data on the blockchain. They ensure that only authorized parties have access to sensitive information.

5.4 LIBRARIES/PACKAGES:

1.Blockchain Platforms:

- Ethereum: Provides a platform for building decentralized applications (DApps) and smart contracts.
- Hyperledger Fabric: A permissioned blockchain framework tailored for enterprise use cases.
- Corda: Designed for businesses to transact directly and privately using smart contracts.

2.Smart Contract Development:

- Solidity: Ethereum's programming language for writing smart contracts.
- Chaincode (Go): Used for developing smart contracts on Hyperledger Fabric.
- Corda Contracts: Provides tools for writing smart contracts on the Corda platform.

3.Cryptographic Libraries:

- CryptoJS: A JavaScript library for cryptographic functions such as hashing and encryption.
- Web3.js: Allows interaction with Ethereum blockchain and smart contracts using JavaScript.
- PyCryptodome: A Python library for cryptographic functions including hashing, encryption, and digital signatures.

4.Consensus Mechanism Libraries:

- Ethereum: Implements its own consensus mechanism (currently transitioning from PoW to PoS).
- Tendermint: A consensus engine that powers various blockchain platforms, including Cosmos and Binance Smart Chain.

5.Encryption Packages:

- OpenSSL: A widely used open-source toolkit for SSL/TLS and cryptographic functions.
- PyCrypto: A Python library providing cryptographic functionalities such as encryption, decryption, and hashing.
- Web Crypto API: A JavaScript API for performing cryptographic operations in web applications.

5.5 Sample Code:

Connect.jsp

```
<%@ page import="java.sql.*"%>
<%@ page import="java.util.*" %>
<%
Connection connection = null;
 try {
                Class.forName("com.mysql.jdbc.Driver");
                connection                                                    =
DriverManager.getConnection("jdbc:mysql://localhost:3306/dias","root","root");
             String sql="";

        }
        catch(Exception e)
        {
                System.out.println(e);
        }
%>
```

Inclusion of Blockchain:

```
<div class="content style1">
   <div class="content_resize">
     <div class="mainbar">
       <div class="article">
         <h2>Upload File</h2>
         <p>&nbsp;</p>
         <div class="clr"></div>
         <span class="style23">
         <% try
          {
                String file=request.getParameter("tt");
                String cont=request.getParameter("text");
                String start=request.getParameter("start");
                String keys = "ef50a0ef2c3e3a5f";
                String h[]=new String[5];
                String filename="filename";
                        String filename1="filename1";
                        String filename2="filename2";
                        String filename3="filename3";
                        String filename4="filename4";
                        ResultSet
rs=connection.createStatement().executeQuery("select * from cloudserver where
fname='"+file+"' ");
                        if(!rs.next())
                        {
                        int sourceFileSize=cont.length();
                        int s=sourceFileSize%4;
                        int s1=sourceFileSize/4;
                        int a1=s1;
                        int a2=s1+s1;
                        int a3=s1+s1+s1;
                        int a4=s1+s1+s1+s1+s;
                        String encryptedValue1="", encryptedValue2="",
encryptedValue3="", encryptedValue4="";
                //      String keys = "ef50a0ef2c3e3a5f";
                                byte[] keyValue = keys.getBytes();
                        Key key = new SecretKeySpec(keyValue, "AES");
                        Cipher c = Cipher.getInstance("AES");
                        c.init(Cipher.ENCRYPT_MODE, key);
                                String encryptedValue = new
String(Base64.encode(cont.getBytes()));

                                encryptedValue1 = new
String(Base64.encode(cont.substring(0,a1).getBytes()));
                                encryptedValue2 = new
String(Base64.encode(cont.substring(a1+1,a2).getBytes()));
                                encryptedValue3 = new
String(Base64.encode(cont.substring(a2+1,a3).getBytes()));
                                encryptedValue4 = new
String(Base64.encode(cont.substring(a3+1,a4).getBytes()));

                        %>
Source File Size = (<%=sourceFileSize %>), a1 = (<%=a1 %>), a2 = (<%=a2 %>), a3
= (<%=a3 %>), a4 = (<%=a4 %>), split : (<%=s %>)
<%
                      PrintStream p1 = new PrintStream(new
FileOutputStream(filename1+".txt"+"/"));
                        p1.print(new String(encryptedValue1));
                        PrintStream p2 = new PrintStream(new
FileOutputStream(filename2+".txt"+"/"));
```

```
                    p2.print(new String(encryptedValue2));
                    PrintStream     p3     =     new     PrintStream(new
FileOutputStream(filename3+".txt"+"/"));
                    p3.print(new String(encryptedValue3));
                    PrintStream     p4     =     new     PrintStream(new
FileOutputStream(filename4+".txt"+"/"));
                    p4.print(new String(encryptedValue4));
                    for(int i=1;i<5;i++)
                    {
                          MessageDigest             md              =
MessageDigest.getInstance("SHA1");
                          FileInputStream      fis11      =      new
FileInputStream(filename+i+".txt"+"/");
                          DigestInputStream dis1 = new DigestInputStream(fis11,
md);
                          BufferedInputStream      bis1      =      new
BufferedInputStream(dis1);

                          //Read the bis so SHA1 is auto calculated at dis
                          while (true)
                          {
                                int b1 = bis1.read();
                                if (b1 == -1)
                                      break;
                          }
                          BigInteger bi1 = new BigInteger(md.digest());
                          String spl1 = bi1.toString();
                          h[i] = bi1.toString(16)}
%>
      </span>
      <div class="clr"></div>
```

Verify Contents of the Block:

```
<%@ page import="java.sql.*" %>
<%@ include file="connect.jsp" %>
       <%
       String owner=(String)application.getAttribute("doname");

             String file = request.getParameter("file");
             String block = request.getParameter("block");
```

```
                try {
                    String mac1,mac2;
                        String query1="";
                        String query2="";
                        if(block.equalsIgnoreCase("Block1"))
                        {
                        query1="select mac1 from backupcloudserver where
fname='"+file+"' and ownername='"+owner+"'";
                        query2="select mac1 from cloudserver where fname='"+file+"'
and ownername='"+owner+"'";
                        }
                        if(block.equalsIgnoreCase("Block2"))
                        {
                        query1="select mac2 from backupcloudserver where
fname='"+file+"' and ownername='"+owner+"'";
                        query2="select mac2 from cloudserver where fname='"+file+"'
and ownername='"+owner+"'";
                        }
                        if(block.equalsIgnoreCase("Block3"))
                        {
                        query1="select mac3 from backupcloudserver where
fname='"+file+"' and ownername='"+owner+"'";
                        query2="select mac3 from cloudserver where fname='"+file+"'
and ownername='"+owner+"'";
                        }
                        if(block.equalsIgnoreCase("Block4"))
                        {
                        query1="select mac4 from backupcloudserver where
fname='"+file+"' and ownername='"+owner+"'";
                        query2="select mac4 from cloudserver where fname='"+file+"'
and ownername='"+owner+"'";
                        }

          Statement st1=connection.createStatement();
          ResultSet rs1=st1.executeQuery(query1);
            if ( rs1.next() )
              {
                    mac1=rs1.getString(1);
                    Statement st2=connection.createStatement();
                ResultSet rs2=st2.executeQuery(query2);
            if ( rs2.next() )
```

```
                {
                        mac2=rs2.getString(1);
                        if(mac1.equalsIgnoreCase(mac2))
                        {
                       // String ss="File Is Safe";
                       // application.setAttribute("fname",file);
                       //application.setAttribute("attacker",ss);
                       %>
<h1> File Named---- <%=file%>  ---- Block Named  --- <%=block %> ---  is
Safe</h1><a href="O_VerifyBlock.jsp">Back</a></br>
                       <%
                        }
                        else
                        {%>
                        <h1> File Named---<%=file%>  ---- Block Named  --- <%=block %>
---is Not Safe</h1><a href="O_VerifyBlock.jsp">Back</a>Do you Want to recover <a
href="recover.jsp?owner=<%=owner%>&block=<%=block%>&file=<%=file%>">R
ecover</a><%
                      //  String ss="attacker";
                       // application.setAttribute("fname",file);
                       //application.setAttribute("attacker",ss);
                        }
                        }}
                        else
                        {%>
                        <h1> File doesnot exist</h1><a href="O_VerifyBlock.jsp">Back</a>
<%
                        }
        }
        catch(Exception e)
        {
                out.println(e.getMessage());
        }
                //response.sendRedirect("O_VerifyBlock1.jsp");
        %>
```

Attack:

```
<%
String aname=request.getParameter("aname");
String file=request.getParameter("fname");
String block=request.getParameter("block");
String owner=request.getParameter("owner");

try
{
String keys = "ef50a0ef2c3e3a5f";
                              String query2="";
                              String cont="";
                              if(block.equalsIgnoreCase("Block-1"))
                              {

                              query2="select ct1 from cloudserver where fname='"+file+"' and
ownername='"+owner+"' ";
                              }
                              if(block.equalsIgnoreCase("Block-2"))
                              {

                              query2="select ct2 from cloudserver where fname='"+file+"' and
ownername='"+owner+"' ";
                              }
                              if(block.equalsIgnoreCase("Block-3"))
                              {

                              query2="select ct3 from cloudserver where fname='"+file+"' and
ownername='"+owner+"'";
                              }
                              if(block.equalsIgnoreCase("Block-4"))
                              {
                                                  query2="select ct4 from cloudserver where
fname='"+file+"' and ownername='"+owner+"' ";
                              }
                              Statement st1=connection.createStatement();
          ResultSet rs1=st1.executeQuery(query2);
                                        if(query2.equalsIgnoreCase("query2"))
                                        {
                                        %>
                              <h1>invalid Query</h1>
```

```
                    <%
                            }
                    ResultSet
rs=connection.createStatement().executeQuery("select * from attacker where
user='"+aname+"' and attacktype='External' ");
                    if(rs.next())
                    {
                    %>
                    <h1>Sorry You are Blocked</h1>
                    <%
                    }
                    else
                    {
                    if(rs1.next())
                    {
                    cont=rs1.getString(1);
                    }
                    else
                    {
                    %>
                    <h2>File Not Exist<h2>
                    <% }
                    byte[] keyValue = keys.getBytes();
                    Key key = new SecretKeySpec(keyValue, "AES");
                    Cipher c = Cipher.getInstance("AES");
                    c.init(Cipher.DECRYPT_MODE, key);
                    String decryptedValue = new
String(Base64.decode(cont.getBytes()));
%>
```

Private Key Generation:

```
<%@ page
import="java.sql.*,java.util.Random,java.security.KeyPair,java.security.KeyPairGener
ator,java.security.NoSuchAlgorithmException,java.security.PublicKey,javax.crypto.Ci
pher,javax.crypto.NoSuchPaddingException" %>
<%@page import ="java.util.*"%>
<%@ page
import="java.sql.*,java.util.Random,java.io.PrintStream,java.io.FileOutputStream,jav
a.io.FileInputStream,java.security.DigestInputStream,java.math.BigInteger,java.securi
ty.MessageDigest,java.io.BufferedInputStream" %>
```

```
<%@                              page                              import
="java.security.Key,java.security.KeyPair,java.security.KeyPairGenerator,javax.crypto
.Cipher"%>
<%@page                                                            import
="java.util.*,java.text.SimpleDateFormat,java.util.Date,java.io.FileInputStream,java.i
o.FileOutputStream,java.io.PrintStream"%>
<%@ include file="connect.jsp" %>
<html><style type="text/css">
<!--
body {
	background-color: #FFFFFF;
}
-->
</style>
<body>
<%

int uid = Integer.parseInt(request.getParameter("uid"));
String fname=request.getParameter("fname");
Statement st1 = connection.createStatement();

try

{
			KeyPairGenerator kg = KeyPairGenerator.getInstance("RSA");
			Cipher encoder = Cipher.getInstance("RSA");
			KeyPair kp = kg.generateKeyPair();
			PublicKey pubKey = kp.getPublic();
			// RSA produces 1024 bits Key
			byte[] pub = pubKey.getEncoded();
			String s = pub.toString();
			String query3 ="update tpadata set sk='"+s+"' where
fname='"+fname+"' ";
			st1.executeUpdate (query3);
			String query4 ="update cloudserver set sk='"+s+"'  where
id='"+uid+"' ";
			st1.executeUpdate (query4);
			String query5 ="update backupcloudserver set sk='"+s+"'  where
fname='"+fname+"' ";
			st1.executeUpdate (query5);
}
catch(Exception e)
{
out.println(e.getMessage());
}
response.sendRedirect("D_ViewFiles1.jsp");
%>
</body>
</html>
```

Recover from an attack:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@page import="java.util.*"%>
<%@ include file="connect.jsp"%>
<%@page
        import="java.util.*,java.security.Key,java.util.Random,javax.crypto.Cipher,jav
ax.crypto.spec.SecretKeySpec,org.bouncycastle.util.encoders.Base64"%>
<%@ page
        import="java.sql.*,java.util.Random,java.io.PrintStream,java.io.FileOutputStr
eam,java.io.FileInputStream,java.security.DigestInputStream,java.math.BigInteger,jav
a.security.MessageDigest,java.io.BufferedInputStream"%>
<%@ page
        import="java.security.Key,java.security.KeyPair,java.security.KeyPairGenerat
or,javax.crypto.Cipher"%>
<%@page
        import="java.util.*,java.text.SimpleDateFormat,java.util.Date,java.io.FileInpu
tStream,java.io.FileOutputStream,java.io.PrintStream"%>
        <%@page
import="com.oreilly.servlet.*,java.sql.*,java.lang.*,java.text.SimpleDateFormat,java.
util.*,java.io.*,javax.servlet.*, javax.servlet.http.*" %>

<%@ page import="java.sql.*"%>

<%@ page import="java.util.*"%>
```

```
<html>
<%
String owner=request.getParameter("owner");
String file=request.getParameter("file");
String block=request.getParameter("block");
String strQuery2="";
String strQuery3="";
String encryptedValue="";
        String          h1="";
        String          pk="";
try
{
                String keys = "ef50a0ef2c3e3a5f";
                        String query2="";
                                if(block.equalsIgnoreCase("Block1"))
                        {
                         strQuery3 = "select mac1,ct1,sk from  backupcloudserver where
fname='" + file + "'  and ownername='"+owner+"' ";
                        }
                                if(block.equalsIgnoreCase("Block2"))
                        {
                                 strQuery3       =       "select      mac2,ct2,sk      from
backupcloudserver where fname='" + file + "'  and ownername='"+owner+"' ";
                        }
                                if(block.equalsIgnoreCase("Block3"))
                        {
                                 strQuery3       =       "select      mac3,ct3,sk      from
backupcloudserver where fname='" + file + "'  and ownername='"+owner+"' ";
                        }
                                if(block.equalsIgnoreCase("Block4"))
                        {
                                 strQuery3       =       "select      mac4,ct4,sk      from
backupcloudserver where fname='" + file + "'  and ownername='"+owner+"' ";
                        }
                        ResultSet
rs=connection.createStatement().executeQuery(strQuery3);
                        if(rs.next())
                        {
                        encryptedValue=rs.getString(2);
                        h1=rs.getString(1);
                        pk=rs.getString(3);
                        }
                                if(block.equalsIgnoreCase("Block1"))
                        {


                                strQuery2 = "update cloudserver set ct1='"
                                                + encryptedValue + "', mac1='" + h1 +
"',sk='" + pk
                                                + "' where fname='" + file + "'  and
ownername='"+owner+"' ";
                        }
                                if(block.equalsIgnoreCase("Block2"))
                        {
                                strQuery2 = "update cloudserver set ct2='"
                                                + encryptedValue + "', mac2='" + h1 +
"',sk='" + pk
                                                + "' where fname='" + file + "'  and
ownername='"+owner+"' ";
                        }
                                if(block.equalsIgnoreCase("Block3"))
                        {

                                strQuery2 = "update cloudserver set ct3='"
                                                + encryptedValue + "', mac3='" + h1 +
"',sk='" + pk
                                                + "' where fname='" + file + "'  and
ownername='"+owner+"' ";
                        }
                                if(block.equalsIgnoreCase("Block4"))
                        {

                                strQuery2 = "update cloudserver set ct4='"
                                                + encryptedValue + "', mac4='" + h1 +
"',sk='" + pk
                                                + "' where fname='" + file + "'  and
ownername='"+owner+"' ";
                        }

        connection.createStatement().executeUpdate(strQuery2);
%>
```

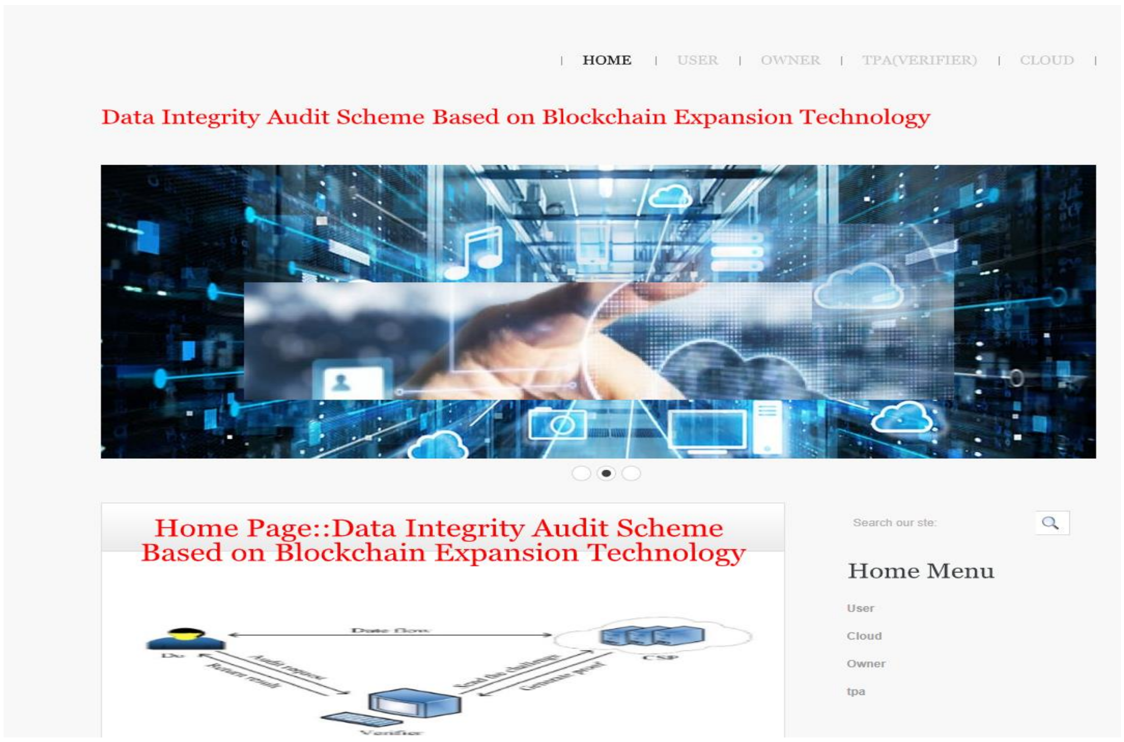
```
<p>
<h1 >Recovered Success Fully!!!<a href="O_VerifyBlock.jsp">Back</a></h1>
</p>
<br />
<%
}
catch (Exception e)
{
e.printStackTrace();
}
%>
</html>
```

6. TESTCASES

Test case ID	**INPUT**	**Expected Output**	**Actual Output**	**Rate**
1.	Modify data and ensure that the hash value changes, indicate tampering.	Indicates a change.	Modifying the data results in a different hash value, indicating a change.	SUCCESS
2.	Test access control mechanis m.	Only authorized users can perform specified tasks.	Only authorized users can perform specific actions (read, write, modify) on data.	SUCCESS
3.	Test the consensus mechanis m to ensure that transaction s are validated.	Validated transaction s should be added to the blockchain .	Valid transaction s are added to the blockchain, and invalid transaction s are rejected.	SUCCESS
4.	Test encryption and decryption functions to ensure the security of data.	Sensitive data should be encrypted.	Sensitive data is encrypted ensuring confidentia lity.	SUCCESS
5.	Verify the accuracy and completen ess of transaction records.	History of transaction s should be immutable and tamper-proof.	Audit logs accurately and completely reflect all data transaction s.	SUCCESS

7. SCREENSHOTS

Home Page: In home page we will be having 5 menu buttons i.e, Home, User, Owner, TPA,Cloud

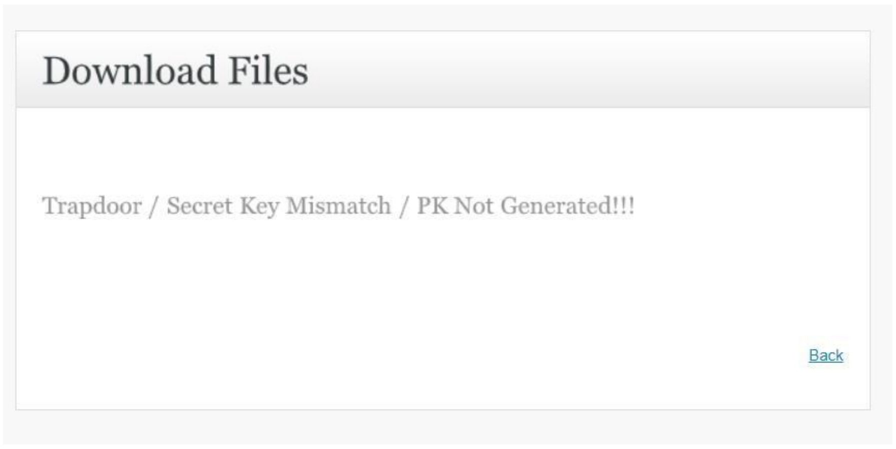


Data Integrity Audit Scheme Based on Blockchain Expansion Technology

Home Page::Data Integrity Audit Scheme Based on Blockchain Expansion Technology

Fig 7.1 Home Page

User:

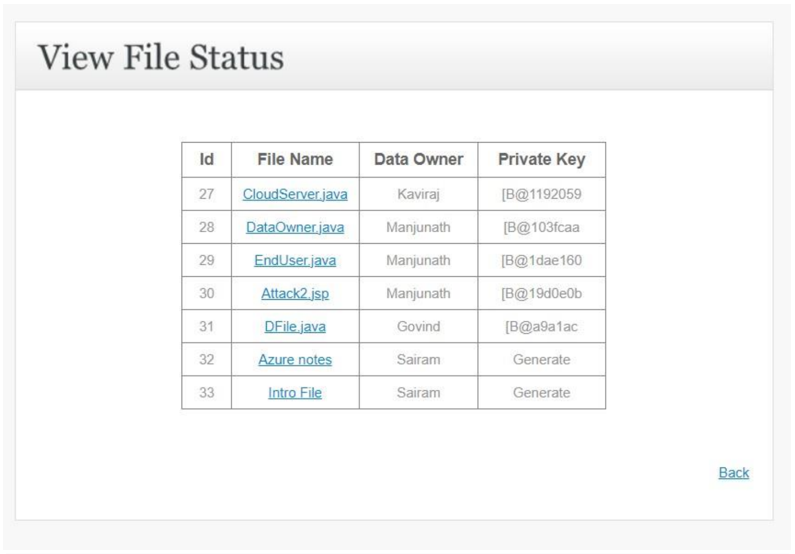


Download Files

Trapdoor / Secret Key Mismatch / PK Not Generated!!!

Fig 7.2 User Page

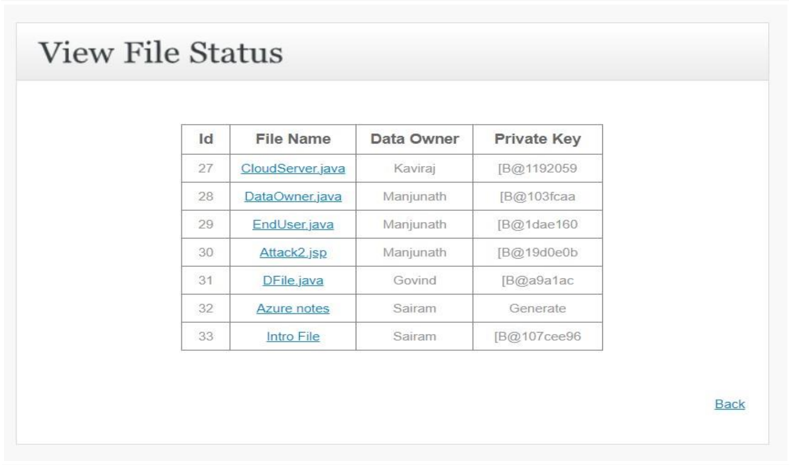
TPA:



View File Status

Id	File Name	Data Owner	Private Key
27	CloudServer.java	Kaviraj	[B@1192059
28	DataOwner.java	Manjunath	[B@103fcaa
29	EndUser.java	Manjunath	[B@1dae160
30	Attack2.jsp	Manjunath	[B@19d0e0b
31	DFile.java	Govind	[B@a9a1ac
32	Azure notes	Sairam	Generate
33	Intro File	Sairam	Generate

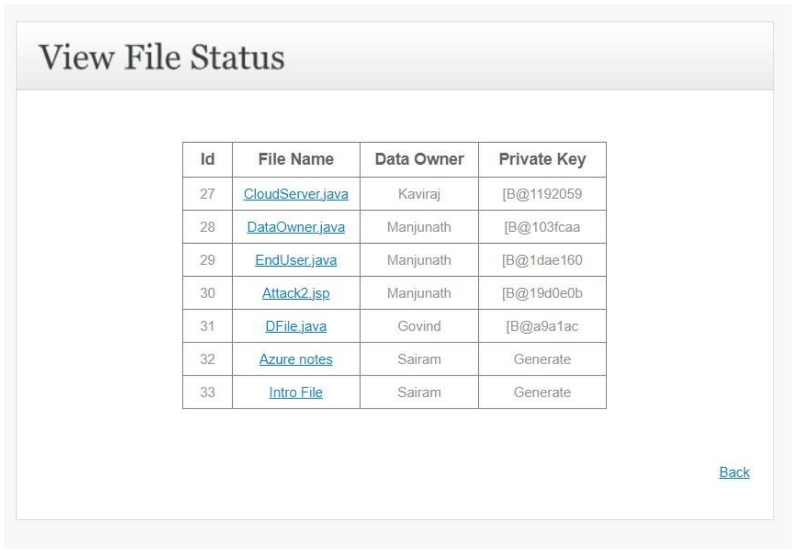
Back



View File Status

Id	File Name	Data Owner	Private Key
27	CloudServer.java	Kaviraj	[B@1192059
28	DataOwner.java	Manjunath	[B@103fcaa
29	EndUser.java	Manjunath	[B@1dae160
30	Attack2.jsp	Manjunath	[B@19d0e0b
31	DFile.java	Govind	[B@a9a1ac
32	Azure notes	Sairam	Generate
33	Intro File	Sairam	[B@107cee96

Back



View File Status

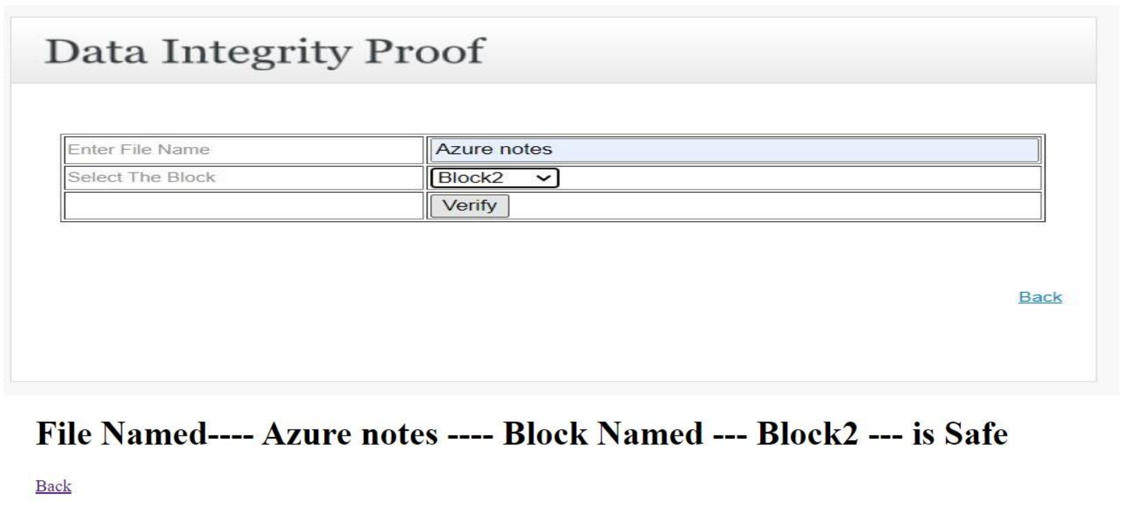
Id	File Name	Data Owner	Private Key
27	CloudServer.java	Kaviraj	[B@1192059
28	DataOwner.java	Manjunath	[B@103fcaa
29	EndUser.java	Manjunath	[B@1dae160
30	Attack2.jsp	Manjunath	[B@19d0e0b
31	DFile.java	Govind	[B@a9a1ac
32	Azure notes	Sairam	Generate
33	Intro File	Sairam	Generate

Back

Fig 7.3 TPA Page

Verifying whether Files block is safe or not

It verifies whether the block is safe or not. If someone tries to modify the block then it display



Data Integrity Proof

Enter File Name	Azure notes
Select The Block	Block2
	Verify

Back

File Named---- Azure notes ---- Block Named --- Block2 --- is Safe

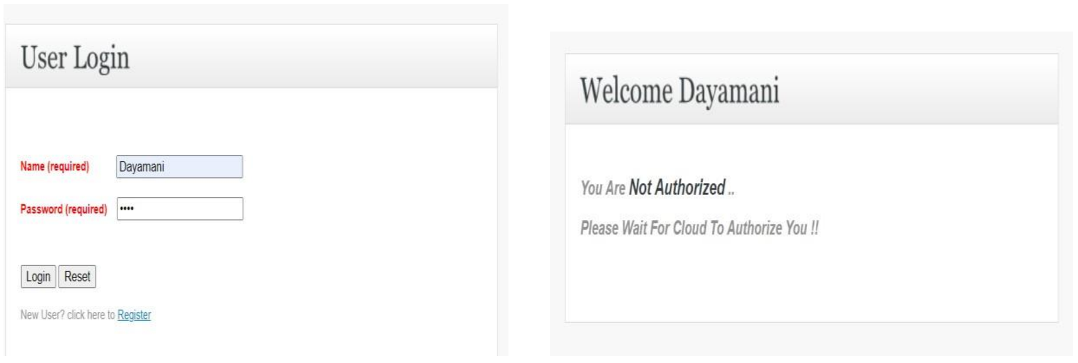
Back

Fig 7.4 Verifying

"**Not Safe**" and the owner can recover

User can perform tasks only if CSP authorizes.

USER:



User Login

Name (required) Dayamani

Password (required) ••••

Login Reset

New User? click here to Register

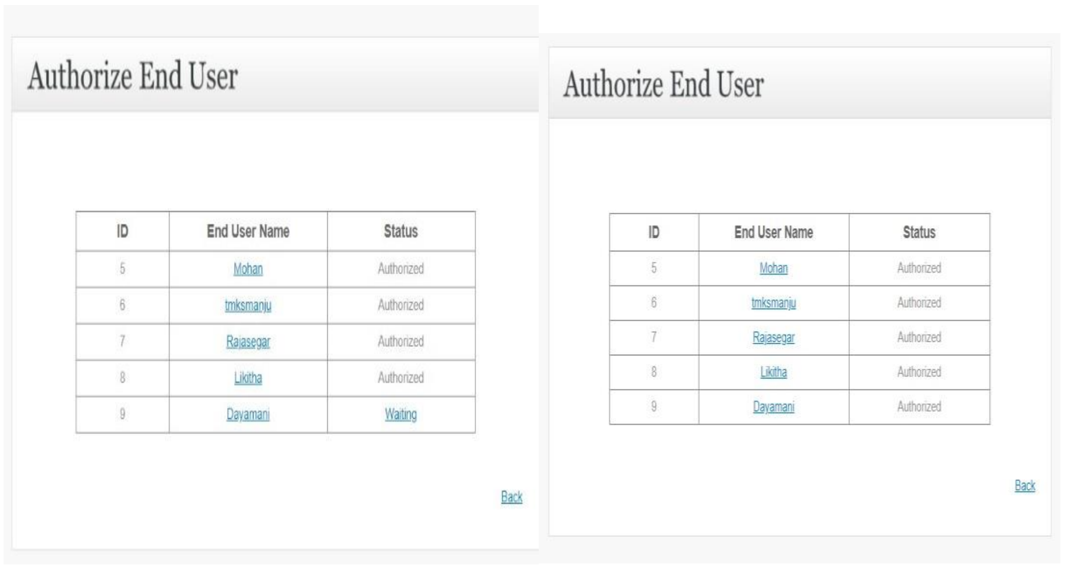
Welcome Dayamani

You Are Not Authorized ..

Please Wait For Cloud To Authorize You !!

Fig 7.5 User Login

CLOUD:



Authorize End User

ID	End User Name	Status
5	Mohan	Authorized
6	tmksmanju	Authorized
7	Rajasegar	Authorized
8	Likitha	Authorized
9	Dayamani	Waiting

Back

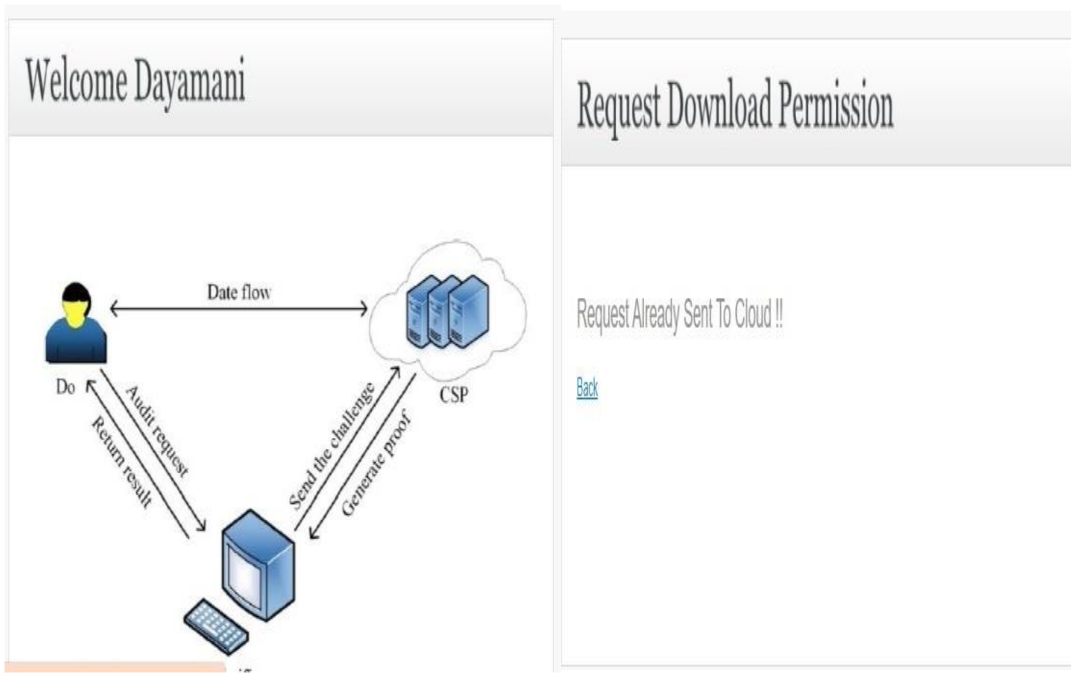
Authorize End User

ID	End User Name	Status
5	Mohan	Authorized
6	tmksmanju	Authorized
7	Rajasegar	Authorized
8	Likitha	Authorized
9	Dayamani	Authorized

Back

Fig 7.6 Cloud Authorization

USER: (When user wants to download)



Welcome Dayamani

Request Download Permission

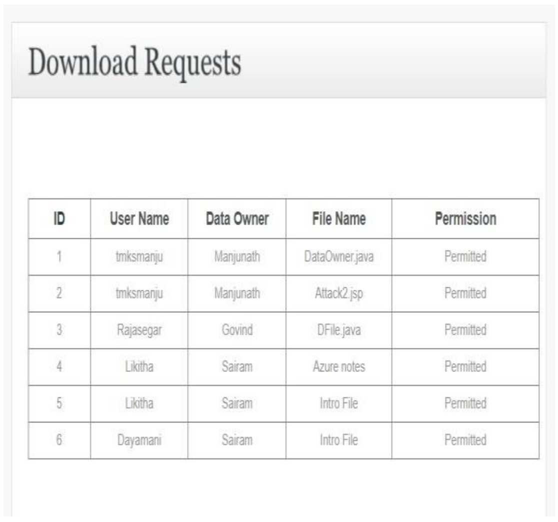
Request Already Sent To Cloud !!

Back

Fig 7.7(a) Download Request

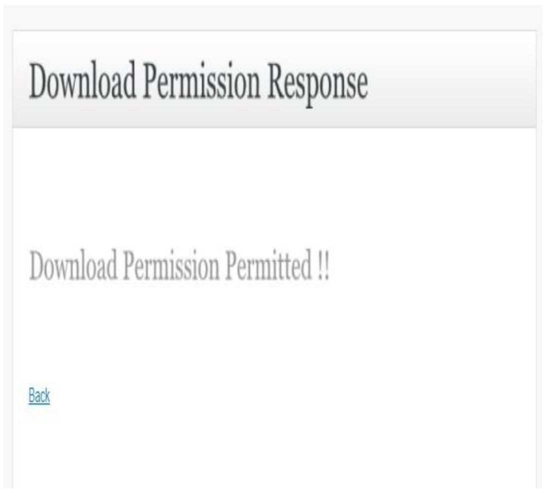
CLOUD: **USER:**

(user can check the responses from the cloud)



Download Requests

ID	User Name	Data Owner	File Name	Permission
1	tmksmanju	Manjunath	DataOwner.java	Permitted
2	tmksmanju	Manjunath	Attack2.jsp	Permitted
3	Rajasegar	Govind	DFile.java	Permitted
4	Likitha	Sairam	Azure notes	Permitted
5	Likitha	Sairam	Intro File	Permitted
6	Dayamani	Sairam	Intro File	Permitted

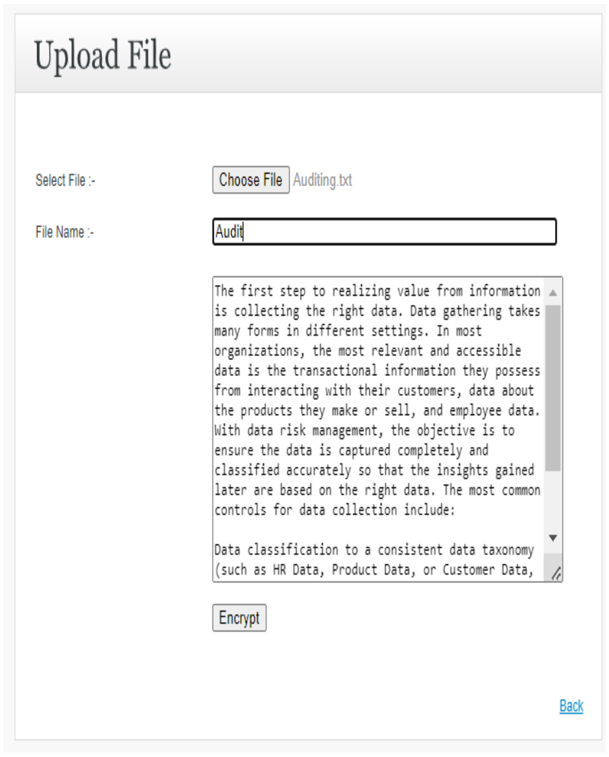


Download Permission Response

Download Permission Permitted !!

Fig 7.7(b) Download Request Granted

Upload Page: In Upload page we will upload the file to the cloud and then we encrypt it.



Upload File

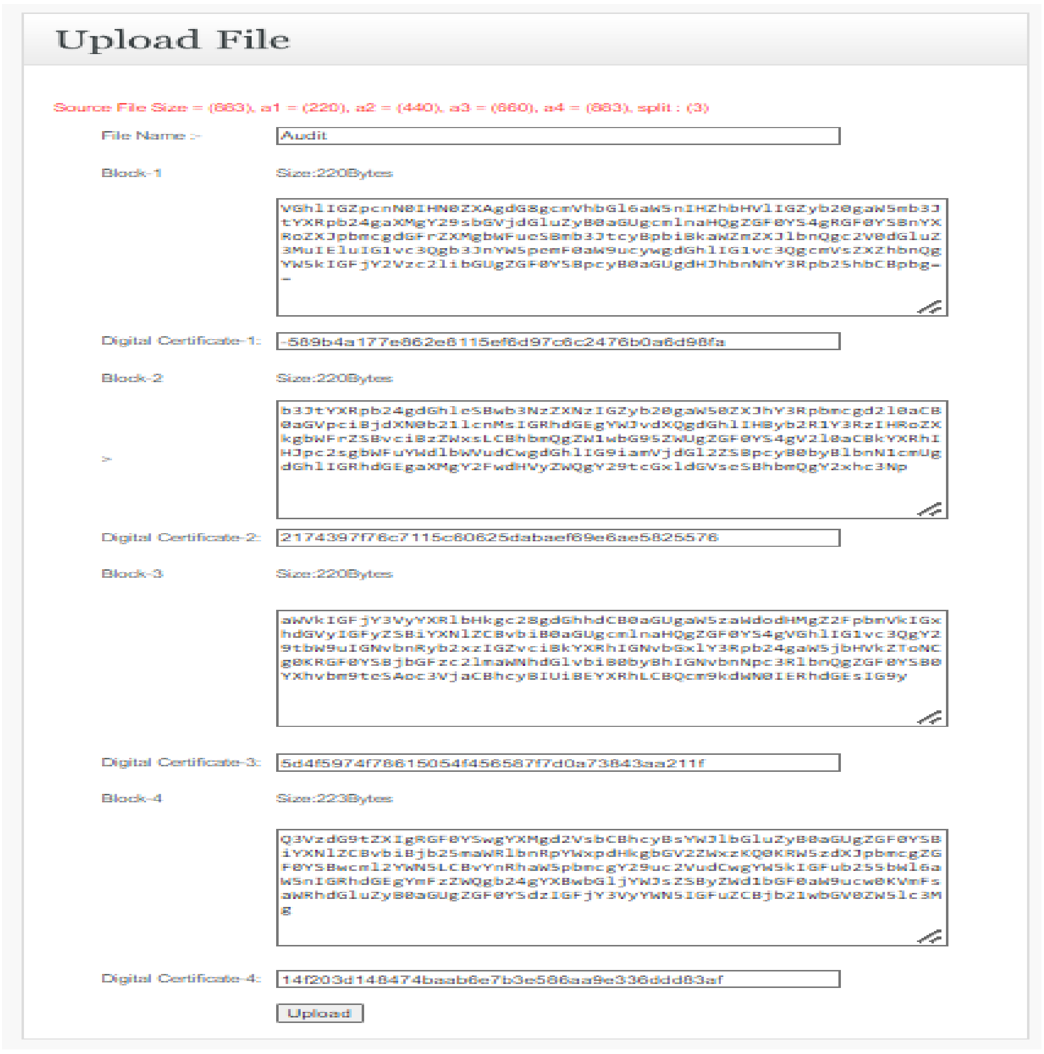
Select File :- Auditing.txt

File Name :- Audit

```
The first step to realizing value from information
is collecting the right data. Data gathering takes
many forms in different settings. In most
organizations, the most relevant and accessible
data is the transactional information they possess
from interacting with their customers, data about
the products they make or sell, and employee data.
With data risk management, the objective is to
ensure the data is captured completely and
classified accurately so that the insights gained
later are based on the right data. The most common
controls for data collection include:

Data classification to a consistent data taxonomy
(such as HR Data, Product Data, or Customer Data,
```

Fig 7.8(a) Upload Page



Upload File

Source File Size = (883), a1 = (220), a2 = (440), a3 = (660), a4 = (883), split : (3)

File Name :-	Audit
Block-1	Size:220Bytes
Digital Certificate-1:	-589b4a177e862e8115ef6d97c6c2476b0a6d98fa
Block-2	Size:220Bytes
Digital Certificate-2:	2174397f76c7115c60625dabaef69e6ae5825576
Block-3	Size:220Bytes
Digital Certificate-3:	5d4f5974f7861505 4f456587f7d0a73843aa211f
Block-4	Size:223Bytes
Digital Certificate-4:	14f203d148474baab6e7b3e586aa9e336ddd83af

```
VGhlIGZpcnN0IHN0ZXAgdG8gcmVhbGl6aW5nIHZhbHVlIGZyb20gaW5mb3J
tYXRpb24gaXMgY29sbGVjdGluZyB0aGUgcmlnaHQgZGF0YS4gRGF0YSBnYX
RoZXJpbmcgdGFrZXMgbWFueSBmb3JtcyBpbiBkaWZmZXJlbnQgc2V0dGluZ
3MuIEluIG1vc3Qgb3JnYW5pemF0aW9ucywgdGhlIG1vc3QgcmVsZXZhbnQg
YW5kIGFjY2Vzc2libGUgZGF0YSBpcyB0aGUgdHJhbnNhY3Rpb25hbCBpbg=
=
```

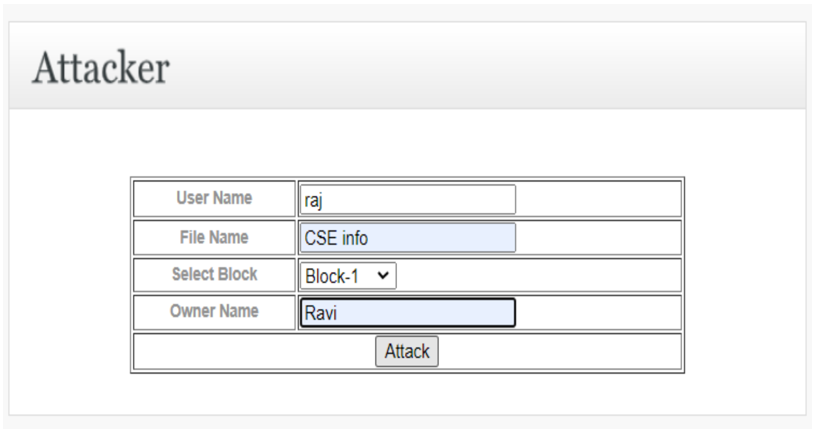
```
b3JtYXRpb24gdGhleSBwb3NzZXNzIGZyb20gaW50ZXJhY3Rpbmcgd2l0aCB
0aGVpciBjdXN0b21lcnMsIGRhdGEgYWJvdXQgdGhlIHByb2R1Y3RzIHRoZX
kgbWFrZSBvciBzZWxsLCBhbmQgZW1wbG95ZWUgZGF0YS4gV2l0aCBkYXRhI
HJpc2sgbWFuYWdlbWVudCwgdGhlIG9iamVjdGl2ZSBpcyB0byB1bnN1cmUg
dGhlIGRhdGEgaXMgY2FwdHVyZWQgY29tcGxldGVseSBhbmQgY2xhc3Np
```

```
aWVkIGFjY3VyYXRlbHkgc28gdGhhdCB0aGUgaW5zaWdodHMgZ2FpbmVkIGx
hdGVyIGFyZSBiYXNlZCBvbiB0aGUgcmlnaHQgZGF0YS4gVGhlIG1vc3QgY2
9tbW9uIGNvbnRyb2xzIGZvciBkYXRhIGNvbGxlY3Rpb24gaW5jbHVkZToNC
g0KRGF0YSBjbGFzc2lmaWNhdGlvbiB0byBhIGNvbnNpc3RlbnQgZGF0YSB0
YXhvbm9teSAoc3VjaCBhcyBIUiBEYXRhLCBQcm9kdWN0IERhdGEsIG9y
```

```
Q3VzdG9tZXIgRGF0YSwgYXMgd2VsbCBhcyBsYWJlbGluZyB0aGUgZGF0YSB
iYXNlZCBvbiBjb25maWRlbnRpYWxpdHkgbGV2ZWxzKQ0KRW5zdXJpbmcgZG
F0YSBwcml2YWN5LCBvYnRhaW5pbmcgY29uc2VudCwgYW5kIGFub255bWl6a
W5nIGRhdGEgYmFzZWQgb24gYXBwbGljYWJsZSByZWd1bGF0aW9ucw0KVmFs
aWRhdGluZyB0aGUgZGF0YSdzIGFjY3VyYWN5IGFuZCBjb21wbGV0ZW5lc3M
g
```

Fig 7.8(b) Upload Page

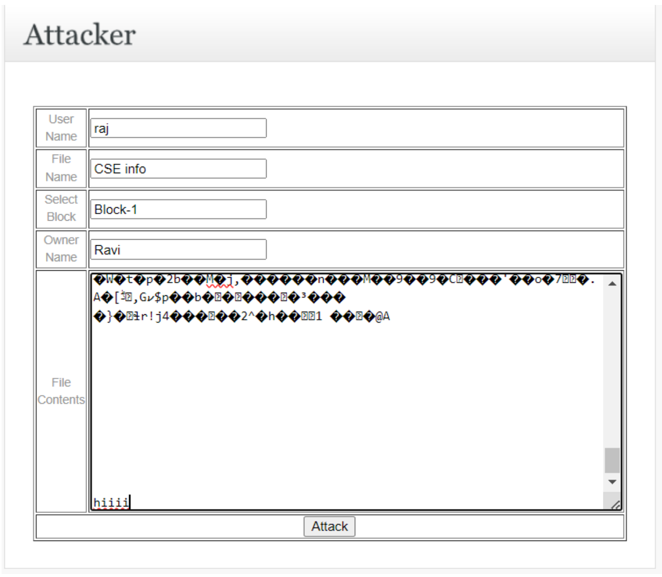
Attacker Page:



Attacker

User Name	raj
File Name	CSE info
Select Block	Block-1
Owner Name	Ravi

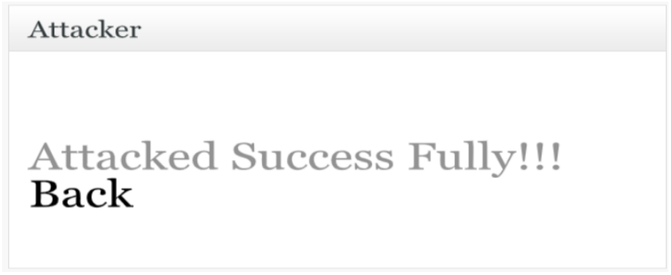
Fig 7.9 Attacker Page



Attacker

User Name	raj
File Name	CSE info
Select Block	Block-1
Owner Name	Ravi
File Contents	�W�t�p�2b��M�j,�������n���M��9���9�C����'��o�7��. A�[�,G⌄$p��b���������³��� �}��r!j4���������2^�h���1 ����@A hiiii

Fig 7.9(a) Attacker Page

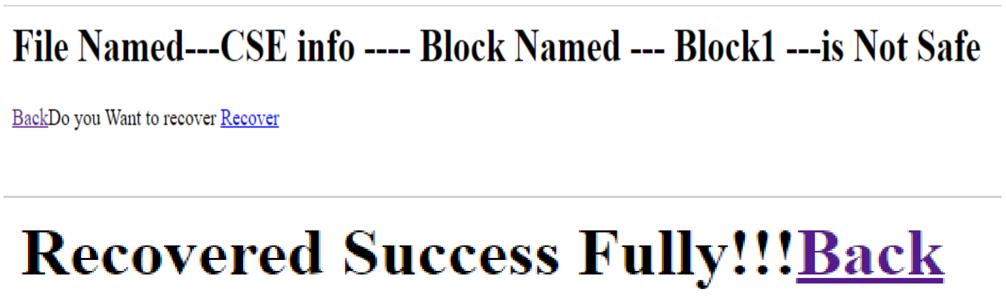


Attacker

Attacked Success Fully!!!
Back

Fig 7.9(b) Attacker Page

Result Page:



File Named---CSE info ---- Block Named --- Block1 ---is Not Safe

BackDo you Want to recover Recover

Recovered Success Fully!!!Back

Fig 7.10 Result Page

8. CONCLUSION

In conclusion, this research addresses the critical societal implications of deepfake technology, specifically focusing on face-mask-altered scenarios during the COVID-19 pandemic. The introduction of the Deepfake Face Mask Dataset (DFFMD) and the innovative approach utilizing Inception-ResNet-v2, Convolutional Neural Networks (CNN), and Xception marks a significant leap forward in detecting manipulated videos. The study's outcomes reveal a notable improvement in accuracy when compared to traditional methods, highlighting the model's effectiveness in identifying deepfakes featuring obscured facial features. As technological advancements continue, this research provides a foundational framework for ongoing improvements, emphasizing the need for robust methods to counteract the potentially malicious use of deepfake technology in our interconnected and information-driven society.

In a rapidly evolving digital landscape, this research holds paramount importance in tackling the growing concerns associated with deepfake technology, especially within the unique context of face-mask-altered scenarios prompted by the global pandemic. The introduction of the Deepfake Face Mask Dataset (DFFMD) and the integration of cutting-edge techniques, such as Inception- ResNet-v2, Convolutional Neural Networks (CNN), and Xception, signify a substantial leap forward in the realm of video manipulation detection. The study's outcomes showcase a remarkable accuracy enhancement over traditional methods, signaling the effectiveness of the proposed model in discerning deepfakes that involve facial obstructions. As technology continues to progress, this research serves as a pivotal milestone, setting the stage for ongoing advancements and emphasizing the urgency of developing resilient methods to combat the potential misuse of deepfake technology in our digitally interconnected society.

9. FUTURE ENHANCEMENTS

1. Enhanced Encryption Mechanisms: Implementing advanced encryption techniques to further enhance the security of data stored in the Cloud. This could involve exploring homomorphic encryption which lets you perform mathematical operations on data without actually decrypting it first.

2. Advanced Analytics and Reporting: Introducing advanced analytics capabilities to provide insights into data usage patterns, access trends, and potential security threats. This could involve implementing data visualization tools and customizable reporting features to empower data owners and administrators with actionable insights.

3. Integration with Emerging Technologies: Exploring integration with emerging technologies such as machine learning or artificial intelligence to enhance data classification, anomaly detection, and threat prevention capabilities. This could involve leveraging AI-driven algorithms to automate data classification and identify potential security vulnerabilities.

4. Enhanced Security Measures: Implementing additional security measures such as multi-factor authentication, biometric authentication, or advanced encryption algorithms to further protect data integrity and user credentials.

5. Improved User Experience: Enhancing the user interface and experience to make it more intuitive, user-friendly, and accessible across different devices and platforms. This could involve redesigning the user interface, optimizing performance, and providing personalized dashboards for users to manage their data effectively.

6. Scalability and Performance Optimization: Optimizing the system architecture and infrastructure to improve scalability, performance, and reliability, especially in handling large volumes of data and concurrent user requests. This may involve implementing distributed computing techniques, caching mechanisms, and load balancing strategies.

7. Cross-Platform Compatibility: Ensuring compatibility and interoperability with various cloud platforms, databases, and operating systems to support a wider range of use cases and deployment scenarios. This could involve standardizing APIs, adopting industry standards, and providing comprehensive documentation and support.

8. Continuous Monitoring and Auditing: Implementing real-time monitoring and auditing capabilities to detect and respond to security incidents, data breaches, and unauthorized access promptly. This could involve integrating security information and event management (SIEM) systems, intrusion detection systems (IDS), and automated alerts/notification mechanisms.

9. Compliance and Regulatory Features: Integrating features to ensure compliance with relevant data protection regulations such as GDPR (General Data Protection Regulation). This could include audit trails for tracking data access and usage, data retention policies, and compliance reporting functionalities.

10. BIBLIOGRAPHY

[1] Y. Fan, X. Lin, G. Tan, Y. Zhang, W. Dong and J. Lei, "One secure data integrity verification scheme for cloud storage", *Future Gener. Comput. Syst.*, vol. 96, pp. 376-385, Jul. 2019.

[2] Q. Zhou, H. Huang, Z. Zheng and J. Bian, "Solutions to scalability of blockchain: A survey", *IEEE Access*, vol. 8, pp. 16440-16455, 2020.

[3] K. Xu, W. Chen and Y. Zhang, "Blockchain-based integrity verification of data migration in multi-cloud storage", *J. Phys. Conf. Ser.*, vol. 2132, no. 1, Dec. 2021.

[4] G. Xie, Y. Liu, G. Xin and Q. Yang, "Blockchain-based cloud data integrity verification scheme with high efficiency", *Secur. Commun. Netw.*, vol. 2021, pp. 1-15, Apr. 2021.

[5] Y. Lei, Z. Jia, Y. Yang, Y. Cheng, and J. Fu, ``A cloud data access authorization update scheme based on blockchain," in *Proc. 3rd Int. Conf.Smart BlockChain (SmartBlock)*, Oct. 2020, pp. 33_38.

[6] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, ``Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362_375, Feb. 2013.

[7] C. Yang, Y. Liu, F. Zhao, and S. Zhang, ``Provable data deletion from ef_cient data integrity auditing and insertion in cloud storage," *Comput. Standards Interface*, vol. 82, Aug. 2022, Art. no. 103629.

[8] H. Yuan, X. Chen, J. Wang, J. Yuan, H. Yan, and W. Susilo, ``Blockchain-based public auditing and secure deduplication with fair arbitration," *Inf.Sci.*, vol. 541, pp. 409_425, Dec. 2020.

[9] L. Chen, Q. Fu, Y. Mu, L. Zeng, F. Rezaeibagha, and M.-S. Hwang, "Blockchain-based random auditor committee for integrity veri_cation," *Future Gener. Comput. Syst.*, vol. 131, pp. 183_193, Jun. 2022.

[10] A. Liu, Y. Wang, and X. Wang, "Blockchain-based data-driven smart customization," in *Data-Driven Engineering Design*. Cham, Switzerland:Springer, 2022, pp. 89_107.