

ENHANCED BLOCKCHAIN BASED DECENTRALIZED PUBLIC AUDITING FOR CLOUD STORAGE

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

By

**PREETHI A (Reg No - 40110999)
DELPHIA J (Reg No – 40110306)**



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

SATHYABAMA

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

**CATEGORY-1 UNIVERSITY BY UGC
Accredited with Grade "A++" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119**

APRIL – 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

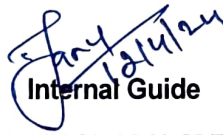
Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Delphia J (40110306)** and **Preethi A(40110999)** who carried out the Project entitled **"ENHANCED BLOCKCHAIN BASED DECENTRALIZED PUBLIC AUDITING" FOR CLOUD STORAGE** under my supervision from November 2023 to April 2024.


Internal Guide

Dr. S. L. JANY SHABU, M.Tech., Ph.D.


Head of the Department
Dr. L. LAKSHMANAN, M.E., Ph.D.
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Juppiar Nagar, Rajiv Gandhi Salai
Chennai - 600 119

Submitted for Viva voce Examination held on 16-04-2024


Internal Examiner


External Examiner

DECLARATION

I, **Delphia J (40110306)** hereby declare that the Project Report entitled **Enhanced Blockchain Based Decentralized Public Auditing for Cloud Storage** done by me under the guidance of **Dr.S.L.JANY SHABU, M.Tech., Ph.D.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 16/04/2024

PLACE : Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

I convey my 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 **Dr. S. L. Jany Shabu M.Tech, Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Cloud computing is emerging as a prevalent data interactive paradigm to realize users' data remotely stored in an online cloud server. Cloud services provide great conveniences for the users to enjoy the on-demand cloud applications without considering the local infrastructure limitations. During the data accessing, different users may be in a collaborative relationship, and thus data sharing becomes significant to achieve productive benefits. The existing PriCloud security solutions mainly focus on the authentication to realize that a user's private data cannot be unauthorized accessed, but neglect a subtle privacy issue during a user challenging the cloud server to request other users for data sharing. Also PriCloud which allows users to pay their storage providers without sacrificing their privacy by employing anonymous storage smart contracts and private payments on a blockchain. The challenged access request itself may reveal the user's privacy no matter whether or not it can obtain the data access permissions. In this paper, we propose a Shared Authority based Privacy-Preserving Authentication Protocol (SAPPAP) to address above privacy issue for cloud storage. In the SAPPAP shared access authority is achieved by anonymous access request matching mechanism with security and privacy considerations (e.g., authentication, data anonymity, user privacy, and forward security). Secondly, attribute based access control is adopted to realize that the user can only access its own data fields; Finally, proxy re-encryption is applied by the cloud server to provide data sharing among the multiple users. Meanwhile, universal composability (UC) model is established to prove that the SAPPAP theoretically has the design correctness. It indicates that the proposed protocol realizing privacy-preserving data access authority sharing, is attractive for multi-user collaborative cloud applications

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 Security-Aware Cloud Architecture	2
2	LITERATURE SURVEY	3
	2.1 Inferences from Literature Survey	5
	2.1.1 Existing System	5
	2.2 Open Problems in Existing System	6
3	REQUIREMENTS ANALYSIS	7
	3.1 Feasibility Studies/Risk Analysis of the Project	7
	3.1.1 Project Scope and Objectives	7
	3.1.2 Technical Feasibility	7
	3.1.3 Legal and Regulatory Compliance	7
	3.1.4 Security and Privacy Measures	8
4	PROPOSED SYSTEM	9
	4.1 Description of Proposed System	9
	4.2 Selected Methodology or Process Model	10
	4.3 Architecture/Overall Design Of Proposed System	11
	4.4 Description of Software for Implementation	12
	4.4.1 Define Scope and Objectives	13
	4.4.2 Data Classification	13
	4.4.3 Access Controls	13
	4.4.4 Data Encryption	13
	4.4.5 Authentication Mechanisms	13
	4.4.6 Logging and Monitoring	13
	4.5 Financial Report on Estimate Costing	13

	4.5.1	Cost Analysis	13
	4.5.2	Data Volumes and Throughput	14
	4.5.3	User Experience and Performance	14
	4.5.4	Privacy-Preserving Technologies	14
	4.5.5	Data Retention and Deletion	14
	4.5.6	Disaster Recovery and Redundancy	14
	4.5.7	Stakeholder and User Acceptance	14
	4.5.8	Environmental Impact	15
	4.5.9	Timeline and Project Plan	15
	4.6	Data Logging	15
	4.6.1	Define Privacy and Security Policies	15
	4.6.2	Access Controls and Authentication	15
	4.6.3	Data Encryption	15
	4.6.4	Monitoring and Logging	16
	4.6.5	Incident Response Plan	16
5		IMPLEMENTATION DETAILS	17
	5.1	Setting up the blockchain network	18
	5.2	Developing smart contracts and API	18
	5.3	Integrating with cloud storage providers	18
6		RESULTS AND DISCUSSIONS	19
7		CONCLUSION	20
8		REFERENCES	21
		APPENDIX	22
		A. SOURCE CODE	22
		B. SCREENSHOTS	71
		C. RESEARCH PAPER	74

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO.
4.1	System Architecture	11
4.2	Implementation and Testing	12
B1	User Dashboard	71
B2	Open Storage	71
B3	Registration	72
B4	Final Output	72

LIST OF ABBREVIATIONS

ABBREVIATIONS

ACO
SHA
GDPA
SAPPAP
TPA

EXPANSIONS

Ant Colony Optimization
Secure Hash Algorithm
General Data Protection Regulation
Shared Authority Based Privacy
Preserving Authentication Protocol
Third Party Administrator

CHAPTER 1

INTRODUCTION

Cloud computing is a promising information technology architecture for both enterprises and individuals. It launches an attractive data storage and interactive paradigm with obvious advantages, including on-demand self-services, ubiquitous network access, and location independent resource pooling. Towards the cloud computing, a typical service architecture is anything as a service (XaaS), in which infrastructures, platform, software, and others are applied for ubiquitous interconnections. Recent studies have been worked to promote the cloud computing evolve towards the internet of services. Subsequently, security and privacy issues are becoming key concerns with the increasing popularity of cloud services. Conventional security approaches mainly focus on the strong authentication to realize that a user can remotely access its own data in on-demand mode. Along with the diversity of the application requirements, users may want to access and share each other's authorized data fields to achieve productive benefits, which brings new security and privacy challenges for the cloud storage.

An example is introduced to identify the main motivation. In the cloud storage based supply chain management, there are various interest groups (e.g., supplier, carrier, and retailer) in the system. Each group owns its users which are permitted to access the authorized data fields, and different users own relatively independent access authorities. It means that any two users from diverse groups should access different data fields of the same file. Thereinto, a supplier purposely may want to access a carrier's data fields, but it is not sure whether the carrier will allow its access request. If the carrier refuses its request, the supplier's access desire will be revealed along with nothing obtained towards the desired data fields. Actually, the supplier may not send the access request or withdraw the unaccepted request in advance if it firmly knows that its request will be refused by the carrier. It is unreasonable to thoroughly disclose the supplier's private information without any privacy considerations.

onto the cloud infrastructure using provider-supported programming languages and software tools (such as Java, Python, or .NET). The user doesn't manage the underlying cloud infrastructure. Popular PaaS platforms include the Google App Engine (GAE) or Microsoft Windows Azure. This level requires securing the provisioned VMs, enforcing security compliance, managing potential risk, and establishing trust among all cloud users and providers.

Securing Software as a Service SaaS employs browser-initiated application software to serve thousands of cloud customers, who make no upfront investment in servers or software licensing. From the provider's perspective, costs are rather low compared with conventional application hosting. SaaS — as heavily pushed by Google, Microsoft, Salesforce.com, and so on — requires that data be protected from loss, distortion, or theft. Transactional security and copyright compliance are designed to protect all intellectual property rights at this level. Data encryption and coloring offer options for upholding data integrity and user privacy.

1.1 SECURITY-AWARE CLOUD ARCHITECTURE

This architecture helps insulate network attacks by establishing trusted operational zones for various cloud applications. Security compliance demands that CSPs protect all data-center servers and storage areas. Our architecture protects VM monitors (or hypervisors) from software-based attacks and safeguards data and information from theft, corruption, and natural disasters. It provides strong authentication and authorized access to sensitive data and on-demand services. We had several design objectives for a trusted and dependable cloud when creating our architecture.

CHAPTER 2

LITERATURE SURVEY

Cong Wang, S.M.chow et al. [1] introduce a privacy preserving public auditing system for cloud data storage. Authors use homomorphic linear authenticator and random masking to prevent TPA to access data content stored on server while the process of auditing, in this way it eliminates the burden of expensive auditing task and fear of data leakage for cloud user.

Most of the times user store their data in private cloud but when extra space is needed user moves towards public cloud, security mechanism in public cloud are there but these are not sufficient to maintain the data integrity. So some other third party mechanism is required for this purpose we use TPA to perform auditing tasks, in this paper author propose an auditing model based on Markel hash tree. This provides a TPA service and store data safely on server.

Ankit R. Mune, R. Pardhi et al. [3] in old days user encrypt data and send it to the server to store data but now days after the emergence of cloud computing data mostly saved on cloud so it to tedious and expensive for user to take extra burden of encrypting data for this purpose author put forward a technique security cloud in which it perform the task of auditing data and maintain the integrity of data keep it safe from any kind of unauthorized access.

Renuka Goyal, Navjot Sidhhu et al. [4] security is the biggest concern in cloud computing, user save their data on cloud, sometimes the is lost or corrupted or discarded by the data center .all these things are the threat for data for this purpose a third party auditor (TPA) is used to take care of all these tasks

Nooper M. et al. [5] to perform data security on cloud storage TPA is used, in this paper A RC5 based Third Party Auditing Mechanism is proposed, that system uses RC5 encryption for data integrity author further extends their results for

multiple user that system provides an highly secure and easy to access third party system.

Nandeesh.B.B et al. [6] in this paper author advanced an dynamic data support for cloud storage including block update, delete, append. It ensure about the correctness of code to provide redundancy parity vector and guarantee the data dependability. It also provide an third party system which can be perform the task of auditing of data and make user worry free about their data.

Jin Li et al [7]. Implemented cloud resource data security model. Cloud Resource handles different cloud services for resource allocation and cloud server resource constraint to satisfy cloud-hosted applications requirements. Effective cloud service depends on the amount of resources available and the type of resources required completing the user's task by each host application. The moment and order of assets assigned is also an aid to the issue of resource allocation. While creating a resource allocation problem, the accessible resources and assigned assets should be detected in the cloud setting.

Lluís Pamies-Juarez et al [8]. implemented multi-objective solution on cloud resources. The objective function of the resource allocation problem can be formulated by using cloud instances, available resources, storage properties and load balancing. Also, static and dynamic resource allocation must be aware of the current state of the server resources in real time scheduling process. Thus, the techniques for monitoring and resource discovery are the important part of this resource allocation system. These requirements are also the inputs for resource optimization algorithms, since it is mandatory to know the services, resources and their condition in order to select those that meet all the cloud specifications.

Deyan Chen et al [9] implemented ACO based elliptic curves and chaotic system to develop a new digital signature algorithm.

Their objectives are analysed and identified, along with empirical validations, pros and cons of each approach.

Boyang Wang et al [10]. proposed a collision resistant one-way hashing on path planning data. In future, further works can be done to enhance some algorithms like MD4, MD5, SHA and, so on. They analysed various pre-existing techniques calculating storage patterns on clouds, which led to some storage and sharing problems. It also resolves some other problems such as data encryption, boundary maintenance and data proof. All these operations are controlled and managed by view management scheme. Some of other advantages of this technique are: entity privacy, data availability and safe data sharing etc. Each and every sensor has its own view and also has a secure boundary.

2.1 INTERFENCES FROM LITERATURE SURVEY

Here in this survey, proposed system offers a technique in which we have reviewed different papers on auditing technique where the batch auditing and single file auditing is performed. The survey discussed about the privacy preserving technique and security approaches while working with the multi copy batch auditing. In this survey we have discussed different security and auditing approach over the cloud and thus the maximum privacy preserving technique to audit the multiple copy verification using efficient signature-based scheme is need to opt out for the further work.

2.1.1 EXISTING SYSTEM

However, most previous researches focus on the authentication to realize that only a legal user can access its authorized data, which ignores the case that different users may want to access and share each other's authorized data fields to achieve productive benefits. When a user challenges the cloud server to request other users for data sharing, the access request itself may reveal the user's privacy no matter whether or not it can obtain the data access permissions. In the earlier work, PriCloud aims to address a user's sensitive access desire related privacy during data sharing in the cloud environments, and it is significant to design a humanistic security scheme to simultaneously achieve data access control, access authority sharing, and privacy.

2.2 OPENPROBLEMS INEXISTINGSYSTEM:

- Previous System does not have the option of granting/revoking data access.
- Malicious cloud service providers may return incorrect integrity audit results.
- Malicious cloud service providers do not comply with customers' requests to completely delete data in the cloud.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/ RISK ANALYSIS

A feasibility study for a privacy-preserving cloud server involves assessing whether it is technically, economically, and operationally viable to implement such a server that prioritizes user privacy. This type of server is critical in the context of data protection and privacy regulations like GDPR and growing concerns about the handling of sensitive information. Here's an outline of the components and considerations to include in your feasibility study:

3.1.1 Project Scope and Objectives:

- Define the purpose and goals of the privacy-preserving cloud server.
- Identify the types of data that need to be stored and processed with privacy in mind.

3.1.2 Technical Feasibility:

- Assess the technical requirements for privacy preservation, such as encryption, access controls, and data anonymization.
- Evaluate the available technology stack and tools that can be used to implement privacy-preserving measures.
- Investigate potential challenges and technical limitations.

3.1.3 Legal and Regulatory Compliance:

- Review applicable data protection laws and regulations (e.g., GDPR, HIPAA) and assess how the server will comply with them.
- Identify the legal constraints and requirements for handling sensitive data.

3.1.4 Security and Privacy Measures:

- Define the security and privacy mechanisms to be implemented, such as end-to-end encryption, data masking, and access auditing.
- Assess the potential risks and vulnerabilities and outline mitigation strategies.

Table3.1: Software Requirement Specification Model

FrontEnd	ASP.NET
Code Language	Visual C#
Database	SQL Server
IDE	Microsoft Visual Studio
Platform	Windows 7 or higher
Browser	Google Chrome or MozillaFirefox

CHAPTER 4

PROPOSED SYSTEM

4.1 DESCRIPTION OF PROPOSED SYSTEM

The address of the mentioned privacy issue to propose a shared authority-based privacy preserving authentication protocol (SAPPAP) for the cloud data storage, which realizes authentication and authorization without compromising a user's private information.

The main contributions are as follows:

- Identify a new privacy challenge in cloud storage, and address a subtle privacy issue during a user challenging the cloud server for data sharing, in which the challenged request itself cannot reveal the user's privacy no matter whether or not it can obtain the access authority.
- Propose an authentication protocol to enhance a user's access request related privacy, and the shared access authority is achieved by anonymous access request matching mechanism.
- Apply cipher text-policy attribute-based access control to realize that a user can reliably access its own data fields, and adopt the proxy re-encryption to provide temp authorized data sharing among multiple users.

A cryptographic solution, with proven security relied on number-theoretic assumptions is more desirable. Data sharing is important functionality in cloud storage. For example, bloggers can let their friends view private data or an enterprise may grant their employee access to important data. But the problem is how to effectively share encrypted data. Of course, users can download the encrypted data from the storage, decrypt them and then send them to others for sharing, but it loses the value of cloud storage. So, user should be able to give access rights of sharing data to others so that they can access these data from server directly.

4.2 SELECTED METHODOLOGY OR PROCESS MODEL

The three layer cloud storage stores in to the three different parts of data parts .If the one data part missing we lost the data information. In this proposed framework using the bucket concept-based algorithms. In our system we using a bucket concept so reduce the data wastages and reduce the process timings.

Steps in Methodology

- Setup Phase
- Keygen Phase
- Encrypt Phase
- Extract Phase
- Decrypt Phase

Setup Phase: Executed by the data owner to setup an account on an untrusted server. On input a security level parameter and the number of ciphertext classes n (i.e., class index should be an integer bounded by 1 and n).

KeyGen Phase: Executed by the data owner to randomly generate a public/master- secret key pair

Encrypt Phase (pk, i, m): Executed by anyone who wants to encrypt data. On input a public-key pk , an index i denoting the ciphertext class, and a message m , it outputs a ciphertext C .

Extract (msk, S): Executed by the data owner for delegating the decrypting power for a certain set of ciphertext classes to a delegate. On input the master secret key msk and a set S of indices corresponding to different classes, it outputs the aggregate key for set S denoted by KS .

On input a security level parameter and the number of ciphertext classes n (i.e., class index should be an integer bounded by 1 and n).

4.3 ARCHITECTURE/OVERALL DESIGN OF PROPOSED SYSTEM

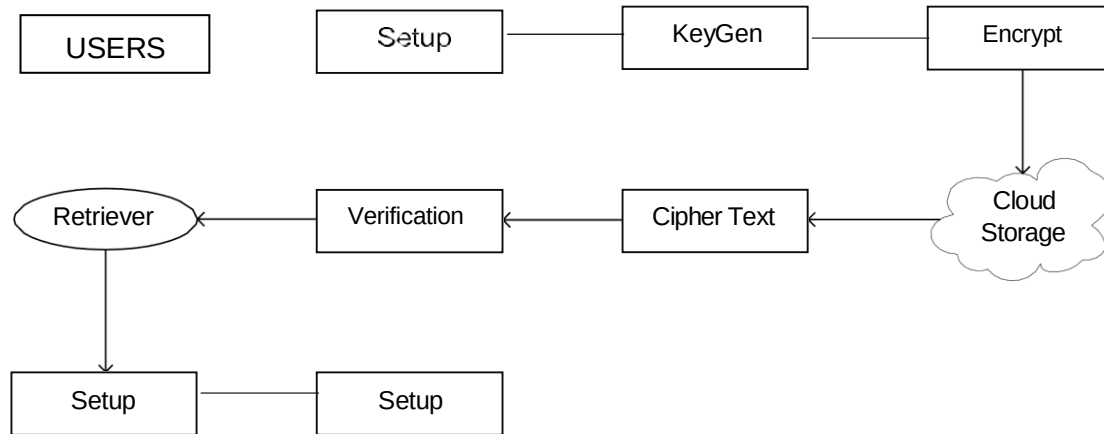


Fig:4.1 : System Architecture

The system architecture is for a proposed system that uses encryption and aggregate keys to store cipher texts in cloud storage. This system involves a setup process, a key generation process, and an encryption process. The system also allows verification of the stored cipher texts using the aggregate key.

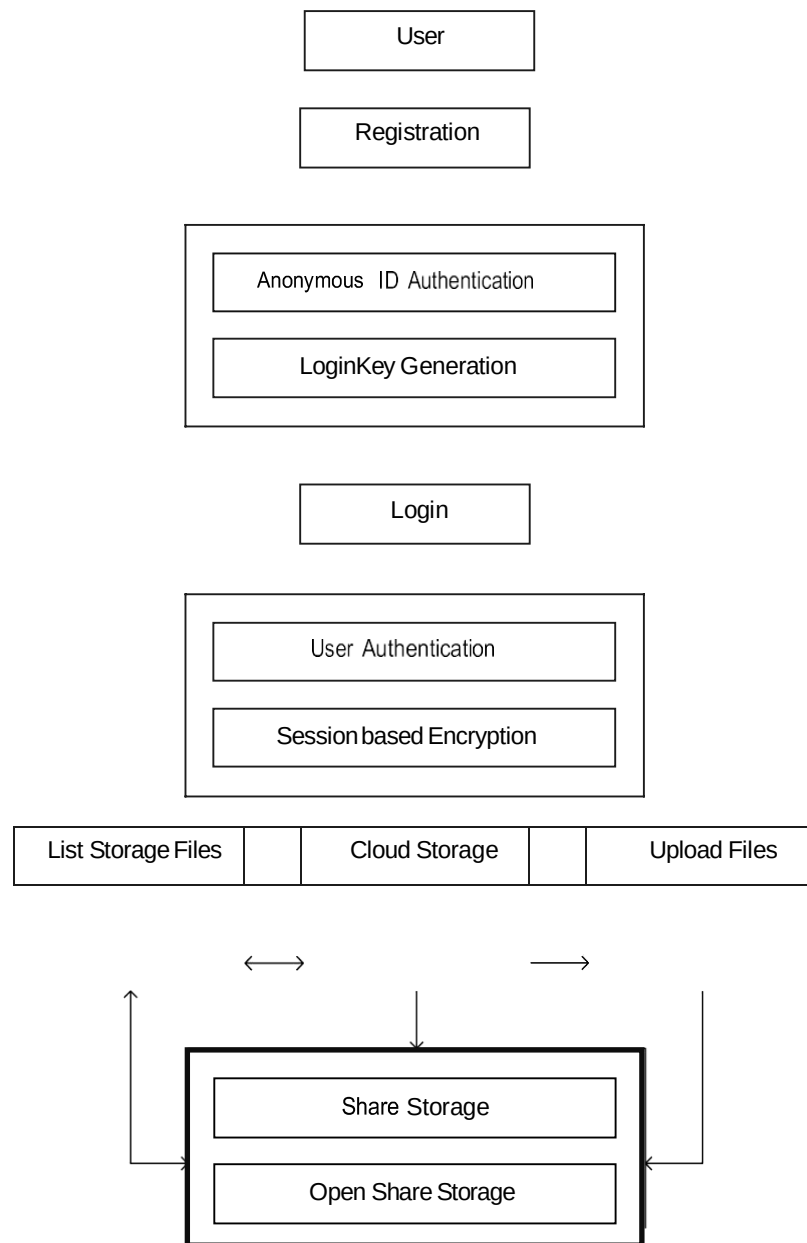


Fig: 4.2 Implementation and Testing

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

Creating a comprehensive testing plan for privacy-preserved cloud servers involves a systematic approach to evaluate various aspects of security and privacy.

4.4.1 Define Scope and Objectives:

- Define the objectives of the testing, ensuring compliance with privacy.

4.4.2 Data Classification:

- Identify and classify the types of data stored on the cloud server based on sensitivity and regulatory requirements.

4.4.3 Access Controls:

- Evaluate user authentication mechanisms.
- Verify role-based access controls and permissions.
- Test for privilege escalation vulnerabilities.

4.4.4 Data Encryption:

- Confirm encryption mechanisms for data in transit and at rest.
- Verify the proper implementation of SSL/TLS protocols.
- Check for encryption key management practices.

4.4.5 Authentication Mechanisms:

- Assess the strength of password policies.
- Test multi-factor authentication.
- Review account lockout policies.

4.4.6 Logging and Monitoring:

- Verify that relevant security events are logged.
- Assess the effectiveness of monitoring tools.
- Set up alerts for suspicious activities.

4.5 FINANCIAL REPORT ON ESTIMATE COSTING

4.5.1 Cost Analysis:

- Estimate the costs associated with implementing and maintaining a privacy-preserving cloud server, including hardware, software, staffing, and ongoing operational costs.
- Compare the cost projections with the expected benefits and cost savings.

4.5.2 Data Volume and Throughput:

- Ensure that the server infrastructure can handle the expected load without compromising privacy or performance.

4.5.3 User Experience and Performance:

- Evaluate the impact of privacy-preserving measures on user experience and system performance.

4.5.4 Privacy-Preserving Technologies:

- Investigate and select the appropriate technologies for privacy preservation, such as secure multi-party computation, homomorphic encryption, or privacy-enhancing protocols.

4.5.5 Data Retention and Deletion:

- Define data retention policies and procedures for securely deleting user data when it is no longer needed.

4.5.6 Disaster Recovery and Redundancy:

- Develop a robust disaster recovery plan and ensure data redundancy for data preservation and privacy during unexpected events.

4.5.7 Stakeholder and User Acceptance:

- Engage with potential users and stakeholders to gather feedback and assess their willingness to use the privacy-preserving cloud server.

4.5.8 Environmental Impact:

- Consider the environmental impact of the server infrastructure and explore sustainable practices.

4.5.9 Timeline and Project Plan:

Develop a project timeline and plan that includes key milestones and deliverables.

4.6 TRANSITION/SOFTWARE TO OPERATIONS PLAN

An operational plan for a privacy-preserved cloud server involves detailing the day-to-day processes, procedures, and guidelines to ensure the secure and private operation of the server. Below is a guide to help you create an operational plan:

4.6.1 Define Privacy and Security Policies:

- Clearly articulate the organization's policies related to data privacy and security.
- Specify the types of data that are considered sensitive and the measures in place to protect them.

4.6.2 Access Controls and Authentication:

- Establish robust access controls, ensuring that only authorized personnel can access the cloud server.

4.6.3 Data Encryption:

- Enforce encryption for data at rest and in transit.
- Regularly review and update encryption protocols to meet industry standards.

4.6.4 Monitoring and Logging:

- Implement comprehensive monitoring tools to track server activities.
- Regularly review logs for any abnormal activities or security incidents.

4.6.5 Incident Response Plan:

- Develop and maintain an incident response plan to address security breaches promptly.
- Conduct regular drills to ensure the effectiveness of the response plan.

CHAPTER 5

IMPLEMENTATION DETAILS

In general, the ASP.NET directory structure can be determined by the developer's preferences. Apart from a few reserved directory names, the site can span any number of directories. The structure is typically reflected directly in the URLs. Although ASP.NET provides means for intercepting the request at any point during processing, the developer is not forced to funnel requests through a central application or front controller.

The special directory names (from ASP.NET 2.0 on) are:

App_Code : This is the "raw code" directory. The ASP.NET server automatically compiles files (and subdirectories) in this folder into an assembly which is accessible in the code of every page of the site. App_Code will typically be used for data access abstraction code, model code and business code. Also any site-specific http handlers and modules and Web service implementation go in this directory. As an alternative to using App_Code the developer may opt to provide a separate assembly with precompiled code.

App_Data : The App_Data ASP.NET Directory is the default directory for any database used by the ASP.NET Website. These databases might include Access (mdb) files or SQL Server (mdf) files. The App_Data is the only directory with Write Access enabled for the ASP.NET web application.

App_LocalResources : E.g. a file called Checkout.aspx.fr-FR.resx holds localized resources for the French version of the Checkout.aspx page. When the UI culture is set to French, ASP.NET will automatically find and use this file for localization.

App_GlobalResources : Holds resx files with localized resources available to every page of the site. This is where the ASP.NET developer will typically store localized messages etc. which are used on more than one page.

App_Themes : Adds a folder that holds files related to themes which is a new ASP.NET feature that helps ensure a consistent appearance throughout a Web site and makes it easier to change the Web site's appearance when necessary.

App_WebReferences : holds discovery files and WSDL files for references to Web services to be consumed in the site.

Bin : Contains compiled code (.dll files) for controls, components, or other code that you want to reference in your application. Any classes represented by code in the Bin folder are automatically referenced in your Application.

5.1 Setting up the Blockchain Network

- Choose a suitable blockchain platform: Evaluate options such as Ethereum, Hyperledger Fabric, or others based on requirements like scalability, interoperability, and consensus mechanisms.
- Deploy the blockchain network: Set up nodes, configure network parameters, and establish connectivity between them.
- Configure network permissions: Define roles and permissions for participants like auditors, cloud service providers, and users.

5.2 Developing Smart Contracts and APIs

- Design smart contracts: Define the structure and logic for auditing processes, including data verification, rewards, and penalties.
- Implement smart contracts: Write code using Solidity (for Ethereum) or other relevant programming languages, ensuring security and efficiency.
- Test smart contracts: Use tools like Truffle or Remix to simulate contract execution and identify bugs or vulnerabilities.
- Develop APIs: Create interfaces for interacting with the blockchain network, allowing users and auditors to submit audit requests, retrieve audit results, and manage accounts.

5.3 Integrating with Cloud Storage Providers

- Identify supported providers: Determine cloud storage platforms to be integrated, such as Amazon S3, Google Cloud Storage, or Microsoft Azure.
- Implement integration modules: Develop connectors or plugins to interact with the cloud storage APIs, enabling data retrieval, verification, and auditing.
- Handle data encryption and decryption: Ensure data security during transit and storage by implementing encryption mechanisms compatible with the blockchain system.
- Test integration: Verify seamless communication between the blockchain network and cloud storage providers, conducting end-to-end testing for functionality and performance.

These implementation steps provide a structured approach to building the blockchain-based public auditing system for cloud storage, encompassing the setup of the blockchain network, development of smart contracts and APIs, and integration with cloud storage providers. Each step should be executed carefully, considering security, scalability, and interoperability requirements.

CHAPTER 6

RESULTS AND DISCUSSIONS

Designing computer output is a critical aspect of system development as it directly impacts the effectiveness of communicating processing results to users. The output must be organized and well-thought-out to ensure that users can easily utilize the system. Effective output design involves creating outputs that are clear, concise, and relevant to the user's needs. It should present information in a structured and visually appealing manner to enhance user understanding and decision-making. By focusing on output design, developers can improve user satisfaction and system usability.

In the realm of cloud computing, privacy challenges have become increasingly prevalent. Issues such as privacy-preserving access authority sharing and data confidentiality have raised concerns about the security of sensitive information stored in the cloud. To address these challenges, cloud service providers must implement robust security measures, such as encryption, access control, and data anonymization, to protect user data from unauthorized access and breaches. Additionally, compliance with data protection regulations, like GDPR, is essential to ensure the privacy and security of user information in the cloud.

Testing principles are fundamental to ensuring the quality and reliability of software systems. Traceability to customer requirements is crucial to verify that the system meets user expectations and functions as intended. Early planning of testing activities allows for the identification of potential issues at an early stage, reducing the cost and effort required for later corrections. Progressive testing, starting from small-scale unit tests to large-scale integration and system tests, helps uncover defects and vulnerabilities across different levels of the software. By following these testing principles, developers can enhance the overall quality of the software and deliver a more robust and reliable product to users.

In conclusion, designing computer output, addressing privacy challenges in cloud computing, and adhering to testing principles are essential components of software development. By focusing on these aspects, developers can create systems that are user-friendly, secure, and high-quality. Effective output design improves user experience, while robust security measures protect sensitive data in the cloud.

CHAPTER 7

CONCLUSION

In conclusion, a new privacy challenge during data accessing in the cloud computing to achieve privacy-preserving access authority sharing. Authentication is established to guarantee data confidentiality and data integrity. Data anonymity is achieved since the wrapped values are exchanged during transmission. User privacy is enhanced by anonymous access requests to privately inform the cloud server about the users' access desires. Forward security is realized by the session identifiers to prevent the session correlation. It indicates that the proposed scheme is possibly applied for enhanced privacy preservation in cloud applications. the implementation of blockchain- based public auditing for cloud storage provides a robust solution for addressing the challenges of data integrity, security, and accountability in cloud storage systems. By leveraging the decentralized and immutable nature of blockchain technology, users can verify the integrity of their data stored in the cloud without relying on a single trusted entity. This enhances transparency, reduces the risk of data manipulation or loss, and fosters greater trust between cloud service providers and their users. However, further research and development are needed to address scalability issues and ensure the practicality of implementing blockchain-based auditing solutions on a large scale. Overall, blockchain-based public auditing holds significant promise for enhancing the security and reliability of cloud storage systems in the future.

REFERENCES

- [1] Ankit R. Mune, R. Pardhi "Security for cloud computing data using a security cloud as a third party auditor (TPA):A Survey" IJARCCCE Vol.3, Issue 3 March 2014.
- [2] Anne Shrijanya K, N. Kashivishwanath "Data Integrity Verification By Third Party Auditor in Remote Data Cloud" IJSCE Vol.3, Issue 5, November 2013.
- [3] Boyang Wang, Baochun Li, Hui Li," Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," IEEE transactions on Cloud Computing, VOL. 2, NO. 01, March 2.
- [4] Cong Wang, S.M. chow, Qian Wang, KuiRen, Wenjing Lou "Privacy Preserving public auditing for Secure cloud storage" IEEE Transaction for Computers Vol.62, No.2, February 2013.
- [5] Deyan Chen, Hong Zhao, "Data Security and Privacy Protection Issues in Cloud Computing", International Conference on Computer Science and Electronics Engineering2012.
- [6] Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P. C. Lee, Wenjing Lou, "A Hybrid Cloud Approach for Secure Authorized Deduplication", IEEE Transactions on Parallel and Distributed Systems, 2014.
- [7] LluísPamies-Juarez, Pedro Garca-Lopez, Marc Sanchez-Artigas, Blas Herrera, "Towards the Design of Optimal Data Redundancy Schemes for Heterogeneous Cloud Storage Infrastructures", Computer Networks, 2011.
- [8] Nandeesh.B.B, Ganesh Kumar R, Jitendranath Mungara" Secure and Dependable Cloud Service for TPA in Cloud computing" IJITEE Vol.1 Issue 3 August 2012.
- [9] Nooper M. Yawale, V.B. Gadichha "Third Party auditing for Data storage Security in cloud withRC5 algorithm" IJARCSSE Vol.3 is Issued on 11 November 2013.
- [10] Renuka Goyal, NavjotSidhhu "Third Party Auditor: An integrity Checking techniquefor client data security inn cloud computing" IJCSIT Vol. 5(3) 2014.
- [11] "Secure and Efficient Proof of Storage with Deduplication in Cloud Computing": Introduces a proof-of-storage system for cloud computing with deduplication, ensuring security and efficiency. (ACM Transactions on Storage, 2014)
- [12] "Toward Secure and Dependable Storage Services in Cloud Computing": Discusses security and privacy issues in cloud storage and proposes solutions for enhanced security. (IEEE Transactions on Services Computing, 2012)

APPENDIX

A.SOURCE CODE

App_code/db.cs

```
using System;

using System.Collections.Generic;using System.Linq;
using System.Web;using System.Data;
using System.Data.SqlClient; using System.Configuration; using
System.Web.UI.WebControls;

/// <summary>
/// Summary description for database
/// </summary>public class db
{
    public static string projtitle = "Privacy Preserved Cloud Server";

    public static SqlConnection con = new
    SqlConnection(ConfigurationManager.ConnectionStrings["dbcon"].ConnectionString);

    public static string GenerateContent(int ContentLength)
    {
        string charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
        string Content = ""; Randomr = new Random();

        for (int i = 0; i <= ContentLength - 1; i++)
        {
            Content += charset[r.Next(0, charset.Length)];
        }
    }
}
```

```

public static Boolean CheckUser()
{
    Boolean Answer = false;
    if (HttpContext.Current.Session["UserType"] == "User")
    {
        Answer = true;
    }
    return Answer;
}

```

```

public static Boolean CheckAdmin()
{
    Boolean Answer = false;
    if (HttpContext.Current.Session["UserType"] == "Admin")
    {
        Answer = true;
    }
    return Answer;
}

```

```

        public static void LoadDropDownList(string Query, DropDownList drpDropDownList,
        Label lblNotification)

```

```

        drpDropDownList.Items.Clear();OpenConnection();
        {
            SqlDataAdapter da = new SqlDataAdapter(Query, con);DataSet ds = new
            DataSet();

            da.Fill(ds);

            if (ds.Tables[0].Rows.Count > 0)
            {
                for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
                {
                    drpDropDownList.Items.Add(ds.Tables[0].Rows[i].ItemArray[0].ToString());
                }
            }
        }
    }
}

```

```
dr.Read();
```

```
if (dr.HasRows) answer = true;
```

```
dr.Close();
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
lblNotification.Text = ex.Message.ToString();
```

```
}
```

```
finally
```

```
{
```

```
con.Close();
```

```
}
```

```
return answer;
```

```
}
```

```
public static Boolean CheckRecord(string Query)
```

```
{
```

```
Boolean answer = false;
```

```
{
```

```
OpenConnection();
```

```
SqlCommand cmd = new SqlCommand(Query, con); SqlDataReader dr =  
cmd.ExecuteReader();
```

```
dr.Read();
```

```
if (dr.HasRows) answer = true;
```

```
dr.Close();
```

```
}
```

```
answer = dr[0].ToString();
```

```
dr.Close();
```

```
}
```

```
catch (Exception ex)
```

```
{
```

```
lblNotification.Text = ex.Message.ToString();
```

```
}
```

```
finally
```

```
{
```

```
con.Close();
```

```
}
```

```
return answer;
```

```
}
```

```
public static int LoadValue(string Query, Label lblNotification)
```

```
{
```

```
int answer = 0;try
```

```
{
```

```
lblNotification.Text = "";OpenConnection();
```

```
OpenConnection();
```

```
SqlCommand cmd = new SqlCommand(Query, con);SqlDataReader dr =  
cmd.ExecuteReader();
```

```
dr.Read();
```

```
if (dr.HasRows) answer = true;
```

```
dr.Close();
```

```
}
```

```
SqlCommand cmd = new SqlCommand(Query, con);SqlDataReader dr =  
cmd.ExecuteReader();
```

```
dr.Read();
```

```
if (dr.HasRows)  
answer = Convert.ToInt32(dr[0].ToString());
```

```
dr.Close();  
}  
catch (Exception ex)  
{  
lblNotification.Text = ex.Message.ToString();  
}  
finally  
{  
con.Close();  
}  
return answer;  
}
```

```
public static string LoadText(string Query)
```

```
{  
string answer = "";try  
{
```

```
OpenConnection();
```

```
SqlCommand cmd = new SqlCommand(Query, con);SqlDataReader dr =  
cmd.ExecuteReader();
```

```
dr.Read();
```

```
answer = dr[0].ToString();
```

```
dr.Close();
```

```
}
```

```
catch
```

```
{
```

```
//do nothing
```

```
}
```

```
finally
```

```
{
```

```
con.Close();
```

```
}
```

```
return answer;
```

```
}
```

```
public static int ExecNonQuery(string Query)
```

```
{
```

```
int Answer = 0;try
```

```
{
```

```
OpenConnection();
```

```
SqlCommand cmd = new SqlCommand(Query, con);int i = cmd.ExecuteNonQuery();
```

```
if (i > 0)
```

```
Answer = 1;
```

```
}
```

```

catch (Exception ex)
{
//donothing
}
finally
{
con.Close();
}
return Answer;
}

```

```

        public static int ExecNonQuery(string Query, Label lblNotification, string
NotificationMessage)
{
int Answer = 0;try
{
lblNotification.Text = "";OpenConnection();
SqlCommand cmd = new SqlCommand(Query, con);int i = cmd.ExecuteNonQuery();

if (i > 0)
lblNotification.Text = NotificationMessage.ToString();

Answer = i;
}
catch (Exception ex)
{
lblNotification.Text = ex.Message.ToString();
}
finally
{
con.Close();
}
return Answer;
}

```

```

public static void LoadGridView(string Query, GridView gvGridView, Label
lblNotification)
{
try
{
gvGridView.DataSource = null;gvGridView.DataBind();

OpenConnection();

SqlDataAdapter da = new SqlDataAdapter(Query, con);DataSet ds = new DataSet();

da.Fill(ds);

if (ds.Tables[0].Rows.Count > 0)
{
gvGridView.DataSource = ds.Tables[0];gvGridView.DataBind();
}
}
catch (Exception ex)
{
lblNotification.Text = ex.Message.ToString();
{
bytesRead = fsIn.Read(buffer, 0, bufferLen);cs.Write(buffer, 0, bytesRead);
} while (bytesRead != 0);

cs.Close();
fsIn.Close();
}

public static byte[] Decrypt(byte[] cipherData, byte[] Key, byte[] IV)
{
MemoryStream ms = new MemoryStream();Rijndael alg = Rijndael.Create(); alg.Key =
Key;

```

```

    }
    finally
    {
        con.Close();
    }
}

public static void LoadDataList(string Query, DataList dlDataList, Label lblNotification)
{
    try
    {
        dlDataList.DataSource = null;dlDataList.DataBind();

        OpenConnection();

        SqlDataAdapter da = new SqlDataAdapter(Query, con);DataSet ds = new DataSet();

        da.Fill(ds);

        if (ds.Tables[0].Rows.Count > 0)
        {
            dlDataList.DataSource = ds.Tables[0];dlDataList.DataBind();
        }
    }
    catch (Exception ex)
    {
        lblNotification.Text = ex.Message.ToString();
    }
    finally
    {
        con.Close();
    }
}

```

App_code/security.cs:

```
using System;
using System.Collections.Generic;using System.Linq;
using System.Web;
using System.Security.Cryptography;using System.Text;
using System.IO;

/// <summary>
/// Summary description for security
/// </summary> public class security
{
    public static byte[] Encrypt(byte[] clearData, byte[] Key, byte[] IV)
    {
        MemoryStream ms = new MemoryStream();Rijndael alg = Rijndael.Create(); alg.Key =
        Key;
        alg.IV = IV;
        CryptoStream cs = new CryptoStream(ms, alg.CreateEncryptor(),
        CryptoStreamMode.Write);cs.Write(clearData, 0, clearData.Length);
        cs.Close();
        byte[] encryptedData = ms.ToArray();

        {
            MemoryStream ms = new MemoryStream();Rijndael alg = Rijndael.Create(); alg.Key =
            Key;
            alg.IV = IV;
            CryptoStream cs = new CryptoStream(ms, alg.CreateDecryptor(),
            CryptoStreamMode.Write);cs.Write(cipherData, 0, cipherData.Length);
            cs.Close();
            byte[] decryptedData = ms.ToArray();return decryptedData;
        }
    }
}
```

```

return encryptedData;
}

```

```

public static string Encrypt(string clearText, string Password)
{
    byte[] clearBytes = System.Text.Encoding.Unicode.GetBytes(clearText);
    PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] { 0x49,
    0x76,
    0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
    byte[] encryptedData = Encrypt(clearBytes, pdb.GetBytes(32), pdb.GetBytes(16));return
    Convert.ToBase64String(encryptedData);
}

```

```

public static byte[] Encrypt(byte[] clearData, string Password)
{
    PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] {
    0x49, 0x76,0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
    return Encrypt(clearData, pdb.GetBytes(32), pdb.GetBytes(16));
}

```

```

public static void Encrypt(string fileIn, string fileOut, string Password)
{

```

```

    FileStream fsIn = new FileStream(fileIn, FileMode.Open, FileAccess.Read); FileStream
    fsOut = new FileStream(fileOut, FileMode.OpenOrCreate, FileAccess.Write);

```

```

    PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] {
    0x49, 0x76,0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });

```

```

    Rijndael alg = Rijndael.Create();alg.Key = pdb.GetBytes(32); alg.IV =
    pdb.GetBytes(16);

```

```

    CryptoStream cs = new CryptoStream(fsOut, alg.CreateEncryptor(),
    CryptoStreamMode.Write);int bufferLen = 4096;
    byte[] buffer = new byte[bufferLen];int bytesRead;

```

```

        PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] {
0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
byte[] decryptedData = Decrypt(cipherBytes, pdb.GetBytes(32), pdb.GetBytes(16));
return System.Text.Encoding.Unicode.GetString(decryptedData);
}

```

```

public static byte[] Decrypt(byte[] cipherData, string Password)
{

```

```

    PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] {
0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
return Decrypt(cipherData, pdb.GetBytes(32), pdb.GetBytes(16));
}

```

```

public static void Decrypt(string fileIn, string fileOut, string Password)
{

```

```

    FileStream fsIn = new FileStream(fileIn, FileMode.Open, FileAccess.Read); FileStream
fsOut = new FileStream(fileOut, FileMode.OpenOrCreate, FileAccess.Write);

```

```

        PasswordDeriveBytes pdb = new PasswordDeriveBytes(Password, new byte[] {
0x49, 0x76, 0x61, 0x6e, 0x20, 0x4d, 0x65, 0x64, 0x76, 0x65, 0x64, 0x65, 0x76 });
Rijndael alg = Rijndael.Create();

```

```

    alg.Key = pdb.GetBytes(32); alg.IV = pdb.GetBytes(16);

```

```

CryptoStream cs = new CryptoStream(fsOut, alg.CreateDecryptor(),
CryptoStreamMode.Write);

```

```

int bufferLen = 4096;
byte[] buffer = new byte[bufferLen]; int bytesRead;

```

```

do
bytesRead = fsIn.Read(buffer, 0, bufferLen); cs.Write(buffer, 0, bytesRead);

```

```
} while (bytesRead != 0);
```

```
cs.Close();  
fsIn.Close();  
}  
}
```

Add cloud storage:

```
using System;  
using System.Collections.Generic;using System.Linq;  
using System.Web; using System.Web.UI;  
using System.Web.UI.WebControls;using System.Text;  
using System.IO; using System.Data;  
using System.Data.SqlClient;  
  
public partial class AddCloudStorage : System.Web.UI.Page  
{  
    protected void Page_Load(object sender, EventArgs e)  
    {  
        catch (Exception ex)  
        {  
            lblNotification.Text = ex.Message.ToString();  
        }  
        finally  
        {  
            con.Close();  
        }  
    }  
}
```

```

if (IsPostBack == false)
{
    if (db.CheckUser() == true)
    {
        UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as
        UpdatePanel; UpdatePanelControlTrigger trigger = new PostBackTrigger();
        trigger.ControlID = btnCreate.UniqueID; updatePanel.Triggers.Add(trigger);

        UpdatePanelControlTrigger triggerupload = new PostBackTrigger();
        triggerupload.ControlID = btnUpload.UniqueID;
        updatePanel.Triggers.Add(triggerupload);

        LoadStorage();
    }
    else
    {
        Response.Redirect("~/Login.aspx");
    }

    void LoadStorage()
    {
        string Query = "select substring(StorageName,1,8) as StorageName from
        tabStorage where UserID='" + Session["UserID"].ToString() + "'";
        db.LoadDataList(Query, dlStorage, lblNotification);

        DataRow dr;
        dt.Columns.Add(new System.Data.DataColumn("StorageName", typeof(String)));

        db.OpenConnection();

        SqlDataAdapter da = new SqlDataAdapter("select StorageName from
        tabStorage where UserID='" + Session["UserID"].ToString() + "'", db.con);
        DataSet ds = new DataSet();
    }
}

```

```

int i = db.ExecNonQuery(Query);

if (i > 0)
{
    lblNotification.Text = "Storage Added Successfully!";LoadStorage();
}
}
else
{
    lblNotification.Text = "Storage Name already exists!";
}
}
catch (Exception ex)
{
    lblNotification.Text = ex.Message.ToString();
}
}
protected void btnUpload_Click(object sender, EventArgs e)
{
    string EncKey;try
    {
        if (fuKeyFile.HasFile)
        {
            StreamReader reader = new StreamReader(fuKeyFile.FileContent);do
            {
                string textLine = reader.ReadLine();EncKey = textLine;
            }
            while (reader.Peek() != -1);

```

```
reader.Close();
```

```
DataTable dt = GetData(EncKey);dlStorage.DataSource = dt; dlStorage.DataBind();  
}  
}  
catch  
{  
Response.Redirect("~/AddCloudStorageInvalidKey.aspx");  
}  
}  
}
```

Add cloud storage invalid key:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/MasterPage.master"  
AutoEventWireup="true" CodeFile="AddCloudStorageInvalidKey.aspx.cs"  
Inherits="AddCloudStorageInvalidKey" %>
```

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">  
</asp:Content>
```

```
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"  
Runat="Server">
```

```
<h1>Incorrect Key File</h1>
```

```
<p>Incorrect Key file entered!</p>
```

```
<p>Try Again?
```

```
<br />
```

```
<asp:Button ID="btnBack" runat="server" Text="Click Here" CssClass="btn"  
onclick="btnBack_Click" /></p>  
</asp:Content>
```

Default.aspx

```

using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{

}
}

```

Download.aspx:

```

using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;

{
string EncKey;try
{
if (fuKeyFile.HasFile)
{
StreamReader reader = new StreamReader(fuKeyFile.FileContent);do
{
string textLine = reader.ReadLine();EncKey = textLine;
}
}
}

```

```

using System.Web.UI.WebControls;using System.IO;

public partial class Download : System.Web.UI.Page
{

private static string FileID; private static string FilePassword;
protected void Page_Load(object sender, EventArgs e)
{
UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as
UpdatePanel;UpdatePanelControlTrigger trigger = newPostBackTrigger();
trigger.ControlID = btnDownload.UniqueID;updatePanel.Triggers.Add(trigger);

if (IsPostBack == false)
{

if (db.CheckUser() == true)
{

using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{

}

}
}

```

```
FileID="" + FileID + "");
```

```
}
```

```
catch
```

```
{
```

```
Response.Redirect("~/ListFiles.aspx", false);
```

```
}
```

```
}
```

```
else
```

```
{
```

```
Response.Redirect("~/Login.aspx");
```

```
}
```

```
}
```

```
}
```

```
static string filepath; static string filename; static Boolean CheckPassword;
```

```
protected void btnDownload_Click(object sender, EventArgs e)
```

```
{
```

```
lblNotification.ForeColor = System.Drawing.Color.Red; lblNotification.Text = "";
```

```
lblNotification.Text = ""; filepath = "~/Files/" + FileID;
```

```
FileInfo file = new FileInfo(Server.MapPath(filepath)); filename =
```

```
Server.MapPath(filepath); lblNotification.Text = filename;
```

```
try
```

```
{
```

```
Response.ContentType = "application/octet-stream"; Response.AddHeader("Content-  
Disposition", "attachment; filename=" +
```

```
lblFileName.Text);
```

```
Response.AddHeader("Content-Length", file.Length.ToString()); Response.ContentType  
= "text/plain";
```

```
Response.TransmitFile(file.FullName);
```

```
lblNotification.Text = "File Downloaded Successfully!"; lblNotification.ForeColor =  
System.Drawing.Color.ForestGreen;  
}  
catch (Exception ex)  
{  
    lblNotification.Text = ex.Message.ToString();  
}  
}  
}
```

Forgot password:

```
using System;  
using System.Collections.Generic;using System.Linq;  
using System.Web; using System.Web.UI;  
using System.Web.UI.WebControls;using System.Text;  
using System.IO;  
  
public partial class ForgotPassword : System.Web.UI.Page  
{  
    protected void Page_Load(object sender, EventArgs e)  
    {  
        if (IsPostBack == false)  
        {  
            UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as  
            UpdatePanel;UpdatePanelControlTrigger trigger = new PostBackTrigger();  
            trigger.ControlID = btnGetPassword.UniqueID;updatePanel.Triggers.Add(trigger);  
        }  
    }  
}
```

```

protected void btnGetPassword_Click(object sender, EventArgs e)
{
    try
    {
        if (fuKeyFile.HasFile)
        {
            StreamReader reader = new StreamReader(fuKeyFile.FileContent);do
            {
                string textLine = reader.ReadLine();EncKey = textLine;
            }
            while (reader.Peek() != -1);reader.Close();
        }

        string UserID = security.Encrypt(txtUserID.Text, EncKey);

        string Query = "select Password from tabUsers where UserID='" + UserID + "'";if
        (db.CheckRecord(Query) == true)
        {
            Session["EncKey"] = EncKey;
            {
                lblNotification.ForeColor = System.Drawing.Color.Red;lblNotification.Text = "";
                lblNotification.Text = ""; filepath = "~/Files/" + FileID;
                FileInfo file = new FileInfo(Server.MapPath(filepath));filename =
                Server.MapPath(filepath); lblNotification.Text = filename;
            }
            try
            {
                Response.ContentType = "application/octet-stream"; Response.AddHeader("Content-
                Disposition", "attachment; filename=" +
                lblFileName.Text);
                Response.AddHeader("Content-Length", file.Length.ToString());Response.ContentType
                = "text/plain";
            }
        }
    }
}

```

```

Query = "select Password from tabUsers where UserID='" + UserID + "'";string
Password = security.Decrypt(db.LoadText(Query), EncKey); lblNotification.Text = "Your
Password is " + Password; lblNotification.ForeColor = System.Drawing.Color.Green;
}
else
{
Response.Redirect("~/LoginFailed.aspx");
}
}
catch (Exception ex)
{
lblNotification.Text = ex.Message.ToString();
}
}
}

```

ListFiles.aspx:

```

using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;using System.Data.SqlClient; using System.Data;

public partial class ListFiles : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
if (IsPostBack == false)
{
if (db.CheckUser() == true)
{
LoadStorage();
LoadFiles();
}
}
}

```

```
else
```

```
{
```

```
Response.Redirect("~/Login.aspx");
```

```
}
```

```
}
```

```
}
```

```
void LoadStorage()
```

```
{
```

```
drpCloudStorage.Items.Clear();try
```

```
{
```

```
db.OpenConnection();
```

```
        SqlDataAdapter da = new SqlDataAdapter("select StorageName from  
tabStorage where UserID='" + Session["UserID"].ToString() + "'", db.con);
```

```
DataSet ds = new DataSet();da.Fill(ds);
```

```
if (ds.Tables[0].Rows.Count > 0)
```

```
{
```

```
string StorageName;
```

```
{
```

```
lblNotification.ForeColor = System.Drawing.Color.Red;lblNotification.Text = "";
```

```
lblNotification.Text = ""; filepath = "~/Files/" + FileID;
```

```
FileInfo file = new FileInfo(Server.MapPath(filepath));filename =
```

```
Server.MapPath(filepath); lblNotification.Text = filename;
```

```
try
```

```
{
```

```
Response.ContentType = "application/octet-stream"; Response.AddHeader("Content-  
Disposition", "attachment; filename=" +
```

```
lblFileName.Text);
```

```
Response.AddHeader("Content-Length", file.Length.ToString());Response.ContentType  
= "text/plain";
```

```

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
{
    StorageName =
security.Decrypt(ds.Tables[0].Rows[i].ItemArray[0].ToString(),
Session["EncKey"].ToString());
drpCloudStorage.Items.Add(StorageName);
}
}
}
catch (Exception ex)
{
    lblNotification.Text = ex.Message.ToString();
}
finally
{
    db.con.Close();
}
}

void LoadFiles()
{
    string Query = "select * from tabFiles where Storage=" +
security.Encrypt(drpCloudStorage.Text, Session["EncKey"].ToString()) + """;
db.LoadGridView(Query, dgvFiles, lblNotification);
}

protected void dgvFiles_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName.Equals("Download"))
    {
        int i = Convert.ToInt32(e.CommandArgument);
        string FileID = dgvFiles.DataKeys[i].Value.ToString();
        Response.Redirect("~/Download.aspx?FileID=" + FileID);
    }
}

```

```
{  
LoadFiles();  
}  
}
```

Login.aspx.cs:

```
using System;  
using System.Collections.Generic;using System.Linq;  
using System.Web; using System.Web.UI;  
using System.Web.UI.WebControls;using System.Text;  
using System.IO;  
  
public partial class Login : System.Web.UI.Page  
{  
protected void Page_Load(object sender, EventArgs e)  
{  
if (IsPostBack == false)  
{  
UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as  
UpdatePanel;UpdatePanelControlTrigger trigger = new PostBackTrigger();  
trigger.ControlID = btnLogin.UniqueID;updatePanel.Triggers.Add(trigger);  
}  
}
```

MasterPage.master.cs:

```
using System;  
using System.Collections.Generic;using System.Linq;  
using System.Web; using System.Web.UI;  
using System.Web.UI.WebControls;  
  
public partial class MasterPage : System.Web.UI.MasterPage  
{  
protected void Page_Load(object sender, EventArgs e)
```

```
}
```

```
public static string EncKey;  
protected void btnLogin_Click(object sender, EventArgs e)  
{  
    Session["EncKey"] = "";try  
    {  
        if (fuKeyFile.HasFile)  
        {  
            StreamReader reader = new StreamReader(fuKeyFile.FileContent);do  
            {  
                string textLine = reader.ReadLine();EncKey = textLine;  
            }  
            while (reader.Peek() != -1);reader.Close();  
        }  
    }
```

```
    string UserID = security.Encrypt(txtUserID.Text, EncKey); string Password =  
    security.Encrypt(txtPassword.Text, EncKey);
```

```
    string Query = "select * from tabUsers where UserID='" + UserID + "' and Password='"  
    + Password + "'";  
    if (db.CheckRecord(Query) == true)  
    {  
        Session["EncKey"] = EncKey;
```

```
        Query = "select UName from tabUsers where UserID='" + UserID + "'";string UserName  
        = security.Decrypt(db.LoadText(Query), EncKey); Session["UserID"] = UserID;  
        Session["UserName"] = UserName; Session["UserType"] = "User";  
        Response.Redirect("~/UserDashboard.aspx");
```

```
    }
```

```
else
```

```
    Response.Redirect("~/LoginFailed.aspx");
```

```

}
catch (Exception ex)
{
    lblNotification.Text = ex.Message.ToString();
}
}
}

```

LoginFailed.aspx.cs:

```

using System;
using System.Collections.Generic; using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;

public partial class LoginFailed : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void btnBack_Click(object sender, EventArgs e)
    {
        Response.Redirect("~/Login.aspx");
    }

    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["UserID"] == null || Session["UserID"].ToString() == "")
        {
            lblUser.Text = "Guest"; lblLogout.Visible = false;

```

```

}
}
}
protected void lbtLogout_Click(object sender, EventArgs e)
{
    Session["UserID"] = "";
    Session["UserType"] = ""; Session.Abandon(); Response.Redirect("~/Login.aspx");
}
}

```

OpenShare.aspx.cs:

```

using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;using System.Data.SqlClient;
{
    lblNotification.ForeColor = System.Drawing.Color.Red;lblNotification.Text = "";
    lblNotification.Text = ""; filepath = "~/Files/" + FileID;
    FileInfo file = new FileInfo(Server.MapPath(filepath));filename =
    Server.MapPath(filepath); lblNotification.Text = filename;
    try
    {
        Response.ContentType = "application/octet-stream"; Response.AddHeader("Content-
        Disposition", "attachment; filename=" +
        lblFileName.Text);
        Response.AddHeader("Content-Length", file.Length.ToString());Response.ContentType
        = "text/plain";
    }
}

```

```
using System.Data;using System.IO;
```

```
public partial class ListFiles : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (IsPostBack == false)
        {
            if (db.CheckUser() == true)
            {
                UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as
                UpdatePanel;UpdatePanelControlTrigger trigger = new PostBackTrigger();
                trigger.ControlID = btnOpenShare.UniqueID;updatePanel.Triggers.Add(trigger);

                UpdatePanelControlTrigger triggerupload = new PostBackTrigger();
                triggerupload.ControlID = btnUpload.UniqueID;
                updatePanel.Triggers.Add(triggerupload);

            }
            else
            {
                Response.Redirect("~/Login.aspx");
            }
        }
        else
        {
            UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as
            UpdatePanel;UpdatePanelControlTrigger triggerupload = new PostBackTrigger();
            triggerupload.ControlID = btnUpload.UniqueID;
            updatePanel.Triggers.Add(triggerupload);
        }
    }

    void LoadFiles(string StorageName)
```

```

string Query = "select * from tabFiles where Storage=" + StorageName + "";
db.LoadGridView(Query, dgvFiles, lblNotification);
}

```

```

protected void dgvFiles_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName.Equals("Download"))
    {
        int i = Convert.ToInt32(e.CommandArgument);
        string FileID = dgvFiles.DataKeys[i].Value.ToString();
        Response.Redirect("~/Download.aspx?FileID=" + FileID);
    }
}

```

```

private static string UploadStorageName = "";private static string UploadKey = "";
private static string UploadPermission = "";
protected void btnOpenShare_Click(object sender, EventArgs e)
{
    try
    {
        string textLine = "";if (fuKeyFile.HasFile)
        {
            StreamReader reader = new StreamReader(fuKeyFile.FileContent);do
            {

```

```

textLine = reader.ReadLine();
}
while (reader.Peek() != -1);reader.Close();
}

string dd = textLine;
string[] separators = { "." };string value = dd.ToString();
string[] words = value.Split(separators, StringSplitOptions.None);

string Permission = words[0].ToString(); string StorageName = words[1].ToString();
string EncKey = words[2].ToString();

UploadStorageName = StorageName;UploadKey = EncKey; UploadPermission =
Permission;
lblStorageName.Text = security.Decrypt(StorageName, EncKey);
LoadFiles(StorageName);
if (Permission == "R")
{
dgvFiles.Columns[3].Visible = true; lblPermission.Text = "Permission : Read";
}
if (Permission == "W")
{
dgvFiles.Columns[3].Visible = true; lblPermission.Text = "Permission : Read &Write";
}
}
catch
{
Response.Redirect("~/OpenShareError.aspx");
}
}

protected void btnUpload_Click(object sender, EventArgs e)
{
if (UploadPermission == "W")

```

```

lblNotification.Text = PostedFile;string allowedChars = "";
allowedChars = "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,";allowedChars +=
"1,2,3,4,5,6,7,8,9,0";

char[] sep = {','};
string[] arr = allowedChars.Split(sep);string fname = "";
string temp = "";

Random rand = new Random();for (int i = 0; i < 20; i++)
{
temp = arr[rand.Next(0, arr.Length)];fname += temp;
}
{
dgvFiles.Columns[3].Visible = true; lblPermission.Text = "Permission : Read";
}
if (Permission == "W")
{
dgvFiles.Columns[3].Visible = true; lblPermission.Text = "Permission : Read &Write";
}
}

```

```

if (fuFile.HasFile)
{
    try
    {
        fuFile.PostedFile.SaveAs(MapPath("~/Files/" + fname));string
        FileName = fuFile.FileName;
        db.con.Open();
        SqlCommand cmd = new SqlCommand("insert into tabFiles
        values('" + fname +
        "','" + fuFile.FileName + "','" + UploadStorageName + "','" +
        FormatFileSize(fuFile.PostedFile.ContentLength) + "','" + Session["UserID"] + "','" +
        DateTime.Now.ToString("dd-MM-yyyy") + "')", db.con);
        cmd.ExecuteNonQuery();
        lblNotification.ForeColor = System.Drawing.Color.Green;lblNotification.Text += "
        Uploaded";
        db.con.Close();
    }
    catch (Exception ex)
    {
        lblNotification.ForeColor = System.Drawing.Color.Red;lblNotification.Text =
        ex.Message.ToString(); db.con.Close();
    }
}
else
{
    lblNotification.Text = "Permission denied.";
}
}

private string FormatFileSize(long bytes)
if (bytes > 1099511627776)

```

```

return ((float)bytes / 1099511627776).ToString("0.00 TB");
}
else if (bytes > 1073741824)
{
return ((float)bytes / 1073741824).ToString("0.00 GB");
}
else if (bytes > 1048576)
{
return ((float)bytes / 1048576).ToString("0.00 MB");
}
else if (bytes > 1024)
{
return ((float)bytes / 1024).ToString("0.00 KB");
}
else return bytes + " Bytes";
}
}

```

OpenShare.Error.aspx.cs:

```

using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;

```

```

public partial class UserDetailsInvalidKey : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{

}
protected void btnBack_Click(object sender, EventArgs e)
{
Response.Redirect("~/OpenShare.aspx");
}
}

```

Registration.aspx.cs:

```

using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;using System.Text;
using System.IO;

public partial class Registration : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
if (IsPostBack == false)
{
UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as
UpdatePanel;UpdatePanelControlTrigger trigger = new PostBackTrigger();
trigger.ControlID = btnRegister.UniqueID;updatePanel.Triggers.Add(trigger);

EncKey = db.GenerateContent(8);

```

```

protected void btnClear_Click(object sender, EventArgs e)
{

}

int SaveFile(string EncKey)
{
    int Answer = 0;try
    {
        string text = EncKey;
        string attachment = String.Format("attachment; filename=" + txtUserID.Text + ".txt");
        HttpContext.Current.Response.ClearHeaders();
        HttpContext.Current.Response.ClearContent();
        HttpContext.Current.Response.AddHeader("content-disposition", attachment);
        HttpContext.Current.Response.ContentType = "application/octet-stream";
        HttpContext.Current.Response.Write(text);
        Answer = 1; HttpContext.Current.Response.End();

    }
    catch (Exception ex)
    {
        {
            dgvFiles.Columns[3].Visible = true; lblPermission.Text = "Permission : Read";
        }
        if (Permission == "W")
        {
            dgvFiles.Columns[3].Visible = true; lblPermission.Text = "Permission : Read &Write";
        }
    }
}

```

```

lblNotification.Text = ex.Message.ToString();
}
return Answer;
}

```

```

private static string EncKey;
protected void btnRegister_Click(object sender, EventArgs e)
{

if (txtUserID.Text.Length > 3)
{
            string Query = "select * from tabUsers where UserID='" + security.Encrypt(
txtUserID.Text, EncKey) + "'";
            if (db.CheckRecord(Query) == false)
            {
                string UName = security.Encrypt(txtName.Text, EncKey); string Gender =
                security.Encrypt(drpGender.Text, EncKey);
                string BirthDate = security.Encrypt(txtBirthDate.Text, EncKey); string MobileNumber =
                security.Encrypt(txtMobileNumber.Text, EncKey); string EmailID =
                security.Encrypt(txtEmailID.Text, EncKey);
                string UserID = security.Encrypt(txtUserID.Text, EncKey); string Password =
                security.Encrypt(txtPassword1.Text, EncKey);

                Query = "insert into tabUsers values('" + UName + "','" + Gender + "','" +
                BirthDate + "','" + MobileNumber + "','" + EmailID + "','" + UserID + "','" + Password + "')";
                int i = db.ExecNonQuery(Query); lblNotification.Text = "Registration Successfull"; if (i >
                0)
                {
                    i = 0;
                    i = SaveFile(EncKey);
                }
                if (i > 0)

```

```

}
else
{
    lblNotification.Text = "User account already exists!";
}
}
else
{
    lblNotification.Text = "User ID is too short";
}
}
}

```

RegistrationSuccess.aspx.cs:

```

using System;
using System.Collections.Generic; using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;

public partial class RegistrationSuccess : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

```

```
}  
}
```

ShareStorage.aspx.cs:

```
using System;  
using System.Collections.Generic;using System.Linq;  
using System.Web; using System.Web.UI;  
using System.Web.UI.WebControls;using System.Data.SqlClient; using System.Data;  
  
public partial class ShareStorage : System.Web.UI.Page  
{  
protected void Page_Load(object sender, EventArgs e)  
{  
if (IsPostBack == false)  
{  
  
UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as  
UpdatePanel;UpdatePanelControlTrigger trigger = new PostBackTrigger();  
trigger.ControlID = btnShare.UniqueID;updatePanel.Triggers.Add(trigger);  
  
UpdatePanelControlTrigger trigger2 = new PostBackTrigger();trigger2.ControlID =  
btnWriteShare.UniqueID; updatePanel.Triggers.Add(trigger2);  
  
if (db.CheckUser() == true)  
{  
  
LoadStorage();  
LoadFiles();  
}  
else  
{
```

```

Response.Redirect("~/Login.aspx");
}
}
}

```

```

void LoadStorage()
{
    drpCloudStorage.Items.Clear();try
    {
        db.OpenConnection();

```

```

                SqlDataAdapter da = new SqlDataAdapter("select StorageName from
tabStorage whereUserID='" + Session["UserID"].ToString() + "'", db.con);
        DataSet ds = new DataSet();da.Fill(ds);
        if (ds.Tables[0].Rows.Count > 0)
        {
            if (db.CheckUser() == true)
            {

```

```

LoadStorage();
LoadFiles();
}
else
{

```

```

string StorageName;
for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
{
    StorageName =
security.Decrypt(ds.Tables[0].Rows[i].ItemArray[0].ToString(),
Session["EncKey"].ToString());
drpCloudStorage.Items.Add(StorageName);
}
}
}
catch (Exception ex)
{
    lblNotification.Text = ex.Message.ToString();
}
finally
{
    db.con.Close();
}
}

void LoadFiles()
{
    string Query = "select * from tabFiles where Owner='" +
Session["UserID"].ToString() + "'and Storage='" +
security.Encrypt(drpCloudStorage.Text, Session["EncKey"].ToString()) + "'";
db.LoadGridView(Query, dgvFiles, lblNotification);
}
int RSaveFile(string EncKey)
{
    int Answer = 0;try
    {
        string FName = EncKey.Substring(0, 8);
        string text = "R." + security.Encrypt(drpCloudStorage.Text,

```

```

Session["EncKey"].ToString()) + "." + Session["EncKey"].ToString();
string attachment = String.Format("attachment; filename=R" + FName + ".txt");
HttpContext.Current.Response.ClearHeaders();
HttpContext.Current.Response.ClearContent();
HttpContext.Current.Response.AddHeader("content-disposition", attachment);
HttpContext.Current.Response.ContentType = "application/octet-stream";
HttpContext.Current.Response.Write(text);
Answer = 1; HttpContext.Current.Response.End();

```

```

}
catch (Exception ex)
{
    lblNotification.Text = ex.Message.ToString();
}
return Answer;
}

```

```

int WSaveFile(string EncKey)
{
    int Answer = 0; try
    {
        string FName = EncKey.Substring(0, 8);
        string text = "W." + security.Encrypt(drpCloudStorage.Text,
        Session["EncKey"].ToString()) + "." + Session["EncKey"].ToString();
        string attachment = String.Format("attachment; filename=W" + FName + ".txt");
        HttpContext.Current.Response.ClearHeaders();
        HttpContext.Current.Response.ClearContent();
        HttpContext.Current.Response.AddHeader("content-disposition", attachment);
        HttpContext.Current.Response.ContentType = "application/octet-stream";
    }
    catch { }
    return Answer;
}

```

```
HttpContext.Current.Response.Write(text);Answer = 1;
HttpContext.Current.Response.End();
```

```
}
catch (Exception ex)
{
    lblNotification.Text = ex.Message.ToString();
}
return Answer;
}
```

```
protected void btnShare_Click(object sender, EventArgs e)
{
    RSaveFile(Session["EncKey"].ToString());
}
protected void btnWriteShare_Click(object sender, EventArgs e)
{
    WSaveFile(Session["EncKey"].ToString());
}
}
```

ShareSuccess.aspx.cs:

```
using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;

public partial class UserDetailsInvalidKey : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void btnBack_Click(object sender, EventArgs e)
```

```
{  
Response.Redirect("~/ShareStorage.aspx");  
}  
}
```

UploadFiles:

```
using System;  
using System.Collections.Generic;using System.Linq;  
using System.Web; using System.Web.UI;  
using System.Web.UI.WebControls;using System.Text;  
using System.IO; using System.Data;  
using System.Data.SqlClient;
```

```
public partial class UploadFiles : System.Web.UI.Page
```

```
if (db.CheckUser() == true)
```

```
{
```

```
LoadStorage();
```

```
LoadFiles();
```

```
}
```

```
else
```

```
{
```

```

{
protected void Page_Load(object sender, EventArgs e)
{
if (IsPostBack == false)
{
if (db.CheckUser() == true)
{
UpdatePanel updatePanel = Page.Master.FindControl("UpdatePanel1") as
UpdatePanel;UpdatePanelControlTrigger triggerupload = new PostBackTrigger();
triggerupload.ControlID = btnUpload.UniqueID;
updatePanel.Triggers.Add(triggerupload);

LoadStorage();
}
else
{
Response.Redirect("~/Login.aspx");
}
}
}

void LoadStorage()
{
drpCloudStorage.Items.Clear();try
{
db.OpenConnection();

        SqlDataAdapter da = new SqlDataAdapter("select StorageName from
tabStorage whereUserID='" + Session["UserID"].ToString() + "'", db.con);
DataSet ds = new DataSet();da.Fill(ds);
if (ds.Tables[0].Rows.Count > 0)
{
string StorageName;

```

```

for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
{
    StorageName =
security.Decrypt(ds.Tables[0].Rows[i].ItemArray[0].ToString(),
Session["EncKey"].ToString());
drpCloudStorage.Items.Add(StorageName);
}
}
}
catch (Exception ex)
{
    lblNotification.Text = ex.Message.ToString();
}
finally
{
    db.con.Close();
}
}
protected void btn Upload_Click(object sender, EventArgs e)
{
    string PostedFile = fuFile.FileName;

    lblNotification.Text = PostedFile;string allowedChars = "";
    allowedChars = "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z,";allowedChars +=
    "1,2,3,4,5,6,7,8,9,0";

```

```

char[] sep = {','};
string[] arr = allowedChars.Split(sep);string fname = "";
string temp = "";

Random rand = new Random();for (int i = 0; i < 20; i++)
{
temp = arr[rand.Next(0, arr.Length)];fname += temp;
}

if (fuFile.HasFile)
{
try
{
fuFile.PostedFile.SaveAs(MapPath("~/Files/" + fname));string
FileName = fuFile.FileName;
db.con.Open();
SqlCommand cmd = new SqlCommand("insert into tabFiles values(" +
fname + "," +
fuFile.FileName + "," + security.Encrypt(drpCloudStorage.Text,
Session["EncKey"].ToString()) + "," + FormatFileSize(fuFile.PostedFile.ContentLength)
+ "," + Session["UserID"] + "," + DateTime.Now.ToString("dd-MM-yyyy") + ")",
db.con);
cmd.ExecuteNonQuery();
lblNotification.ForeColor = System.Drawing.Color.Green;lblNotification.Text += "
Uploaded";
db.con.Close();
}
catch (Exception ex)
{
lblNotification.ForeColor = System.Drawing.Color.Red;lblNotification.Text =
ex.Message.ToString(); db.con.Close();
}
}
private string FormatFileSize(long bytes)

```

```

{
if (bytes > 1099511627776)
{
return ((float)bytes / 1099511627776).ToString("0.00 TB");
}
else if (bytes > 1073741824)
{
return ((float)bytes / 1073741824).ToString("0.00 GB");
}
else if (bytes > 1048576)
{
return ((float)bytes / 1048576).ToString("0.00 MB");
}
else if (bytes > 1024)
{
return ((float)bytes / 1024).ToString("0.00 KB");
}
else return bytes + " Bytes";
}
}
if (db.CheckUser() == true)
{

LoadStorage();
LoadFiles();
}
else
{

```

UserDashboard.aspx.cs:

```
using System;
using System.Collections.Generic;using System.Linq;
using System.Web; using System.Web.UI;
using System.Web.UI.WebControls;

public partial class UserDashboard : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (IsPostBack == false)
        {
            if (db.CheckUser() == true)
            {
                string Query = "select * from tabUserMenu order by 1";db.LoadDataList(Query, dlMenu,
                lblNotification);
            }
            else
            {
                Response.Redirect("~/Login.aspx");
            }
        }
        protected void dlMenu_ItemCommand(object source, DataListCommandEventArgs e)
        {
            if (e.CommandName == "OpenLink")
            {
                string MenuID = dlMenu.DataKeys[Convert.ToInt32(e.CommandArgument)].ToString();
                string NavURL = db.LoadText("select NavURL from tabUserMenu where MenuID='" +
                MenuID.ToString() + "'");
                Response.Redirect(NavURL);
            }
        }
    }
}
```

B. SCREENSHOTS

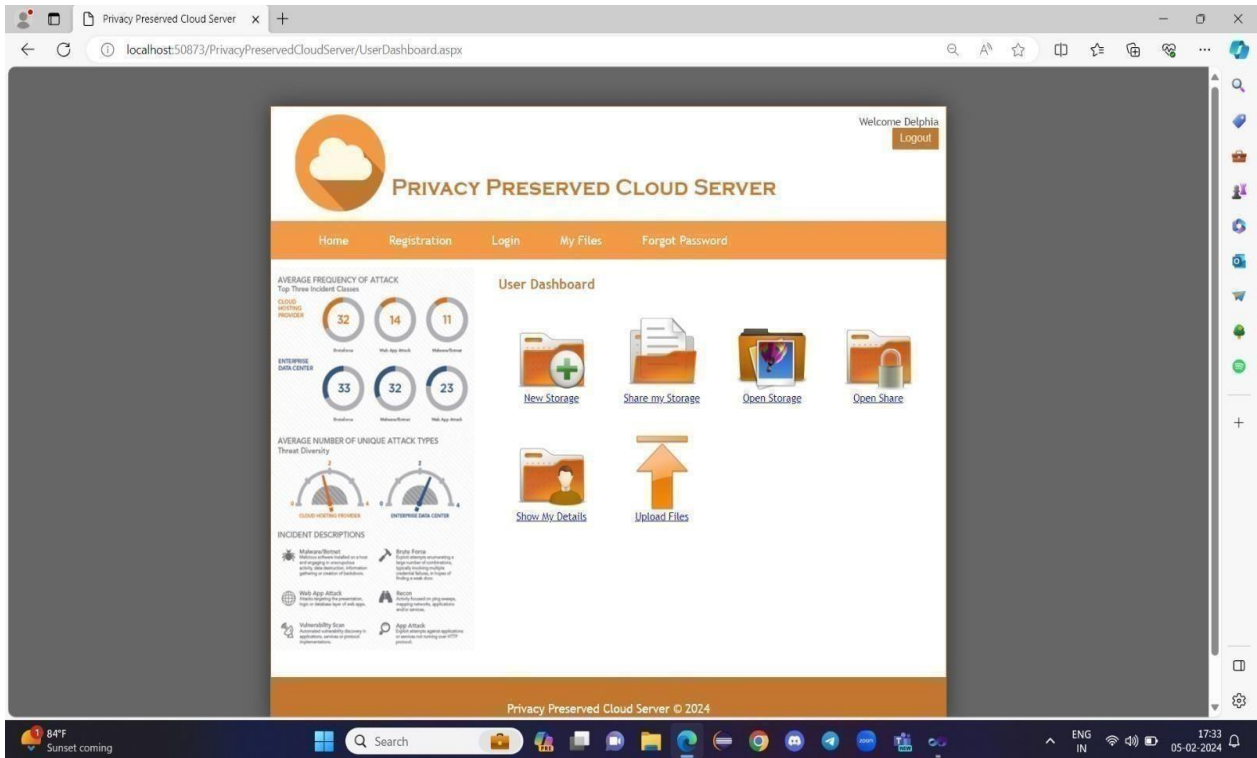


Fig B1: User Dashboard

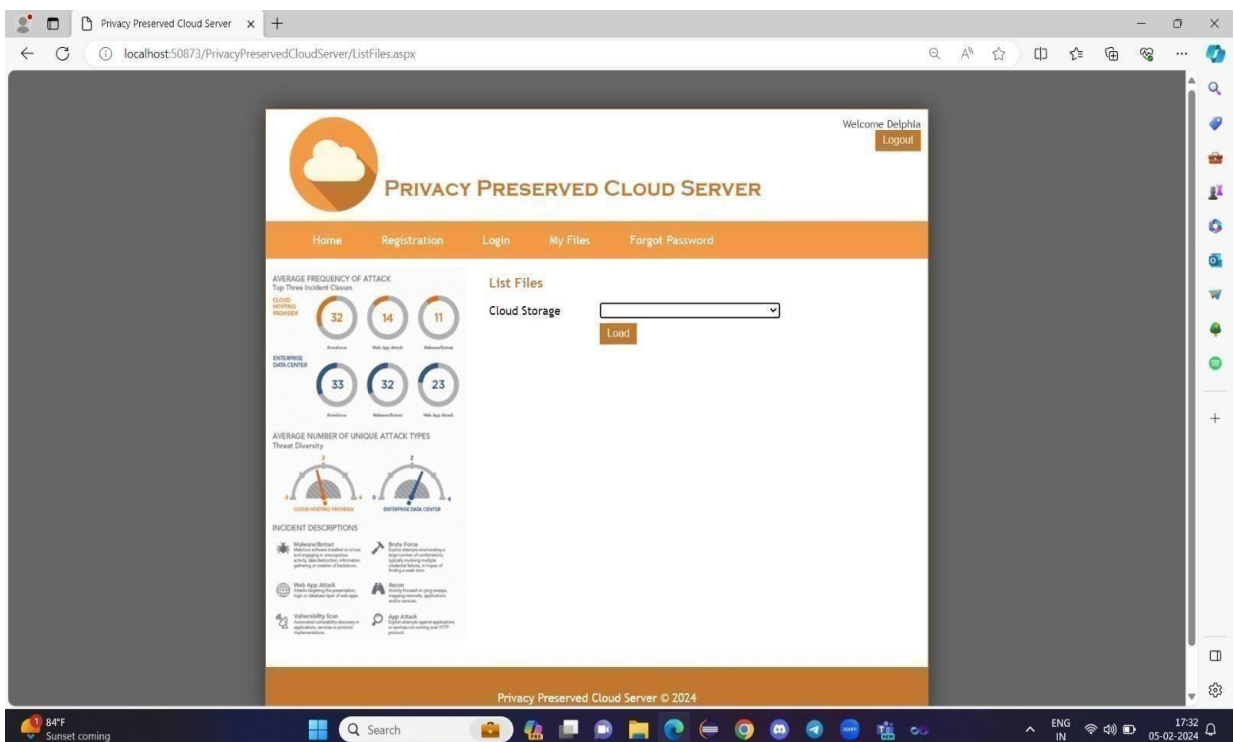


Fig B2: Open Storage

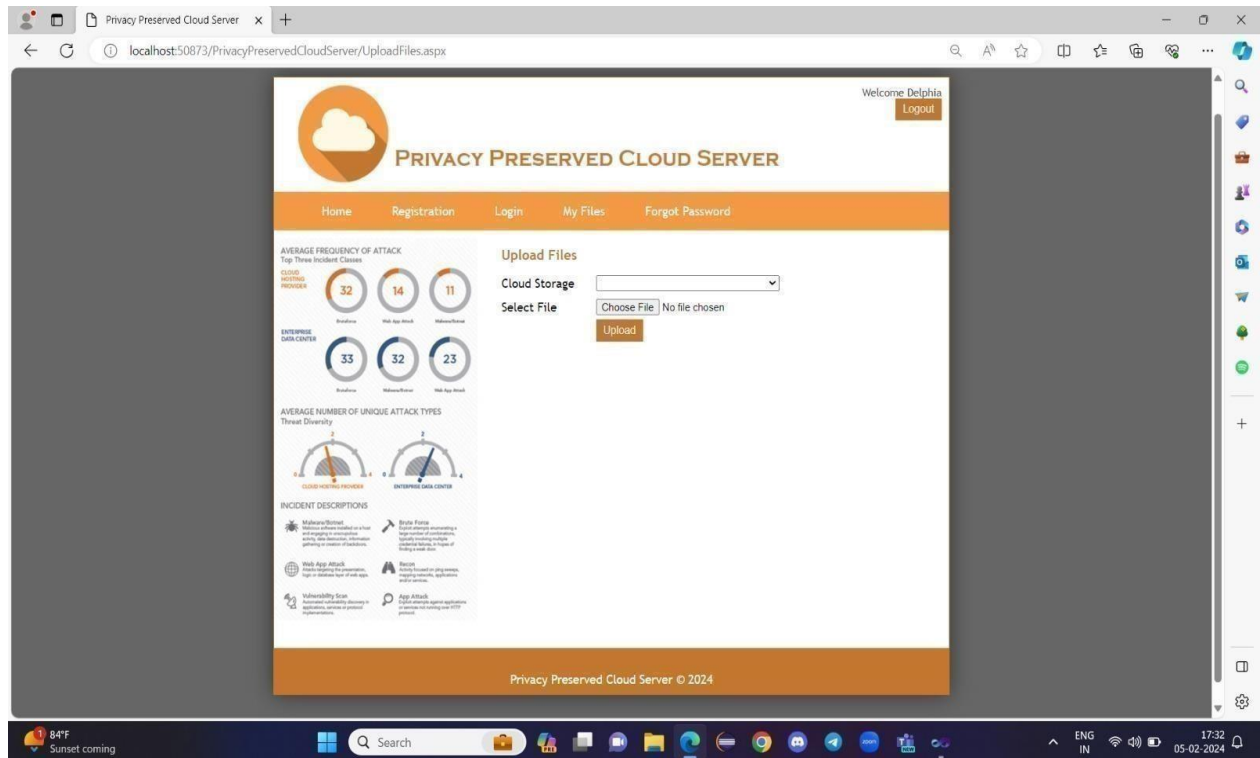


Fig B3: Registration

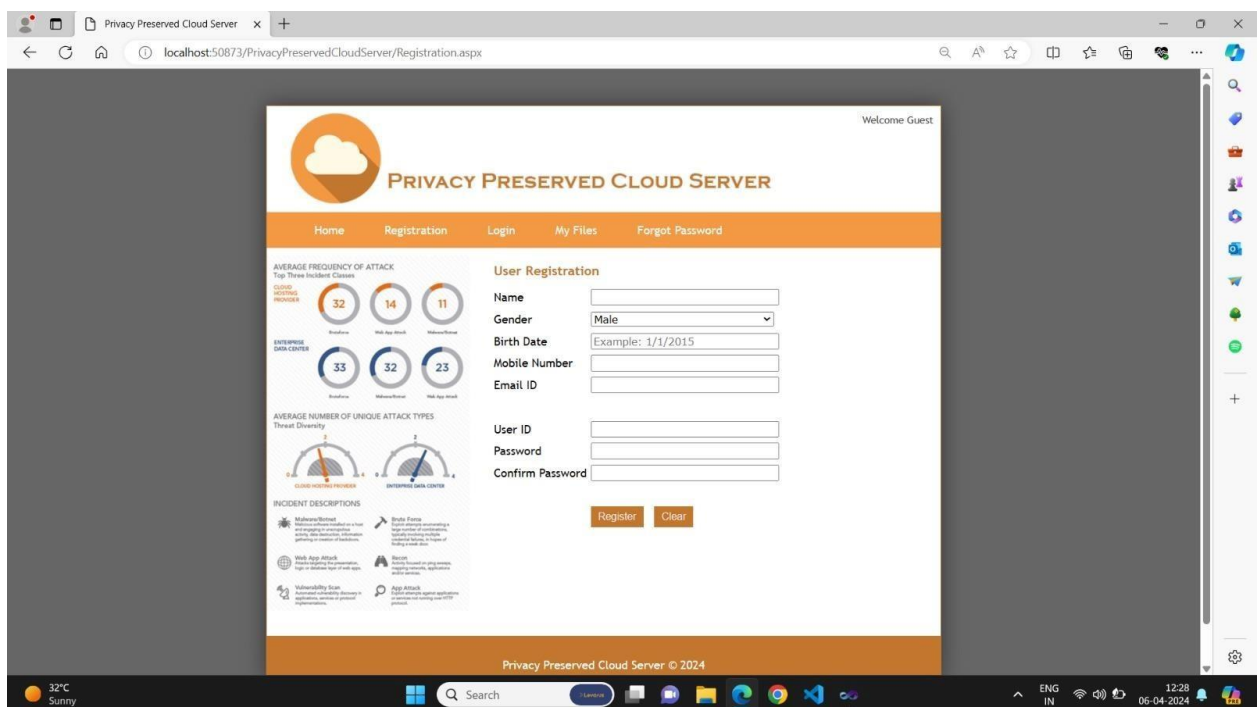


Fig B3: Final Output

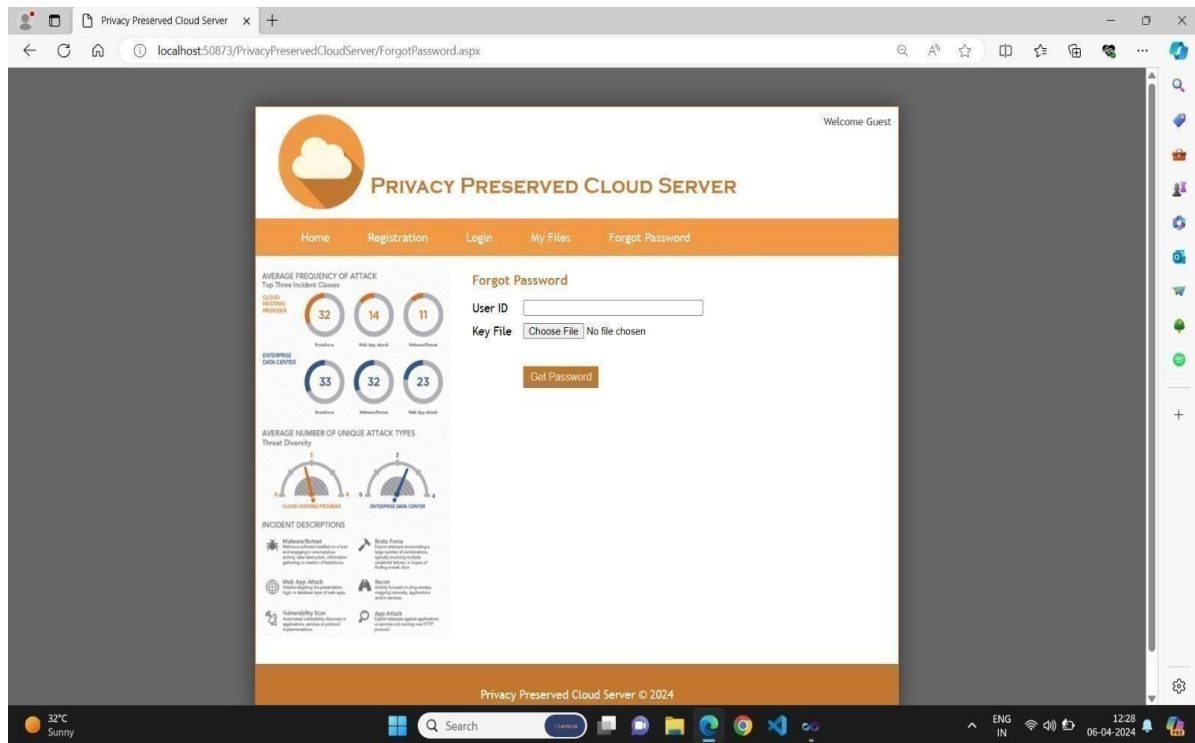


Fig B4: Forgot Password

Enhanced Blockchain Based Decentralized Public Auditing for Cloud Storage

Preethi.A

*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
preethi522002@gmail.com*

Delphia J

*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
delphiadel7@gmail.com*

S.L Jany Shabu

*Department of Computer Science and
Engineering
Sathyabama Institute of Science and
Technology
Chennai, India
Janyshabu.cse@sathyabama.ac.in*

Abstract — Considering the development of cloud computing, customers' data may now be accessed remotely from any location via an online cloud server. Thanks to cloud services, users can experience the great ease of on-demand cloud apps, which don't take into account local infrastructure constraints. Since various users may be working together while accessing the data, data sharing becomes essential to achieving beneficial outcomes. In order to ensure that a user's private data cannot be read by unauthorized parties, the current PriCloud security solutions mostly concentrate on authentication. They overlook a small privacy issue, though, which occurs when a user asks the cloud server to look for ways to share data with other users. Additionally, there is PriCloud, which uses blockchain-based private payments and anonymous storage smart contracts to let customers pay their storage providers without giving up their privacy. Whether or not the challenged access request is successful in obtaining the necessary data access rights, it may nevertheless disclose the user's privacy. To address the aforementioned privacy issue for cloud storage, we propose in this work a Privacy-Preserving Authentication Protocol with Shared Authority (PPAPS). The matching method for anonymous access requests in the PPAPS allows for shared access authority while taking security and privacy into account. The second stage guarantees a user can access just their own information fields by implementing attribute-based access control. Lastly, the cloud server uses proxy re-encryption to facilitate sharing of data between different users.

Keywords: Cloud Security, Privacy Preserved Authentication, Cloud Authentication

I. INTRODUCTION

A potential information technology architecture for businesses and individuals alike is cloud computing. It introduces a compelling interaction and data storage paradigm with clear benefits, such as location-independent resource pooling, omnipresent network

connectivity, and on-demand self-services. Anything as a service (XaaS) is a common service architecture in cloud computing, where infrastructures, platforms, software, and other elements are applied for widespread interconnections. In order to assist cloud computing's transition to the internet of services, recent research has been done. As cloud services gain popularity, security and privacy concerns rise to the top of the priority list. Traditionally, security methods have focused mostly on strong authentication to allow users to access their own data remotely and whenever they want. This raises more concerns about cloud storage privacy and security.

By providing an example, the main motivation is shown. Users who are allowed to access the approved data fields belong to each group, and various users own somewhat autonomous access authorities. It implies that distinct data fields inside the same file should be accessible to any two users from different groups. Therefore, even if a supplier knows that it wants access to a carrier's data fields, there's no guarantee the carrier will agree to the request. The supplier's access request and any information pertaining to the sought data fields will be made public if the carrier rejects the request.

1.1 System Model

The three primary network components in cloud storage architecture are:

1. User
2. Cloud Server
3. Trusted Third-Party Auditor (TPA)

Users can use cloud services applications, platforms, or online infrastructures to store data remotely. Cloud services operate in distributed, cooperative, and parallel modes. When a user accesses cloud data, they can communicate with the cloud server independently and independently of others. They are also given complete control over their own data fields. The system uses decentralized key management system, which enhance the security system, so that even the cloud owner cannot access the data owner's data. The decentralized key management also increase the performance of the storage

server, by using encryption and decryption in the client side.

II. LITERATURE SURVEY

C. Castelluccia et al [1], in order to perform aggregation operations on the ciphertexts, the author has created a symmetric key homomorphic encryption technique. The technique is effective for CPU-limited devices, such as WSN sensor nodes, because it makes use of modular addition. Additionally, their approach is capable of computing mean, variance, and deviation—all statistical values—in an efficient manner. But because they employed symmetric homomorphic encryption, their aggregator was able to decrypt the data from each individual sensor, and they presumptively included a reliable aggregator in their model.

According to Qi et al. [2], author provides customized network services via a variety of network terminals and access points; ubiquity and mobility aspects should be taken into account. These capabilities are available in today's high-tech mobile phones, sometimes known as smartphones. In actuality, the majority of smartphones have improved internet capabilities, enabling the quickest internet connections—3G and 4G. Therefore, the adoption of cloud computing in the healthcare industry and other service provider sectors can be made easier when such mobile phones are readily available in a nation. This will make it simpler to connect to data, information, and computer resources from any location at any time, facilitating communication between healthcare actors such as patients, doctors, and healthcare providers.

T. Jung et al. [4], uses the assumption that every protocol communication channel is unreliable. The discrete logarithm will satisfy: 1) the huge prime numbers are the ordering of the integer groupings; 2) there are enormous amounts of integer numbers involved. Their scheme's security depends on this supposition. They also presuppose the existence of a secure pseudorandom function (PRF) that, in terms of computing, is able to choose a random element from a group in a manner similar to uniform random.

M. Joye et al. [5], proposed a method to handle massive amounts of unencrypted data from numerous users. This paper tackles the issue of private data analysis, wherein an external party must calculate aggregate statistics over personally identifiable sensitive data. This work offered a unique technique that allows the total of a user's private inputs to be assessed by an untrusted aggregator. There are no limitations on the number of users or message space, unlike previous methods.

III. PROPOSED SYSTEM

The proposed address the previously mentioned privacy concern in this research and provide a Privacy Preserving Authentication Protocol with Shared Authority (PPAPS) for cloud data storage that achieves permission and authentication without jeopardizing a user's personal data.

The following are the primary contributions:

1. Respond to a small privacy concern that comes up when someone asks a cloud server to share their data. Even if

the challenged request is successful in gaining access authority, it cannot by itself disclose the user's privacy. This offers cloud storage users a new privacy danger.

2. Provide an authentication protocol to protect user privacy connected to access requests. An anonymous access request matching method is used to accomplish shared access authority.

3. To ensure that a user may consistently access its own data fields, utilize attribute-based cipher text-policy access control. Use proxy re-encryption thereafter to enable transient multi-user data exchange.

ENHANCED PRIVACY PRESERVED CLOUD SERVICES

Achieving privacy in a cloud server involves implementing various security measures to protect data from unauthorized access, whether it's from malicious actors or even the cloud service provider itself.

Encryption: Encrypt data both in transit and at rest. Use strong encryption algorithms to ensure that even if data is intercepted, it remains unreadable. Tools like SSL/TLS for transit encryption and AES encryption for data at rest are commonly used.

Access Control: Implement strict access controls to limit who can access the data and what actions they can perform. Use techniques like role-based access control (RBAC) to assign permissions based on roles and responsibilities.

Strong Authentication: Enforce strong authentication mechanisms such as multi-factor authentication (MFA) to prevent unauthorized access, even if credentials are compromised.

Data Minimization: Only store data that is necessary for operations, and regularly purge any unnecessary data. This reduces the potential impact in case of a breach.

Regular Audits and Monitoring: Monitor user activities and regularly audit access logs to detect any suspicious behavior or unauthorized access attempts.

Network Security: Implement firewalls, intrusion detection/prevention systems, and other network security measures to protect the cloud server from external threats.

Data Residency and Jurisdiction: Choose cloud service providers that offer options for data residency and jurisdiction to ensure that data is stored in locations that comply with relevant privacy regulations and laws.

Secure Development Practices: Follow secure development practices when building and deploying applications on the cloud server to minimize vulnerabilities that could be exploited by attackers.

Regular Software Updates and Patch Management: Keep the cloud server's operating system, applications, and security software up to date with the latest patches and updates to address known vulnerabilities.

Data Encryption Key Management: Properly manage encryption keys to ensure that they are secure and only accessible to authorized users.

Secure Data Transfer: Ensure that data transferred to and from the cloud server is done securely using protocols like HTTPS or SFTP.

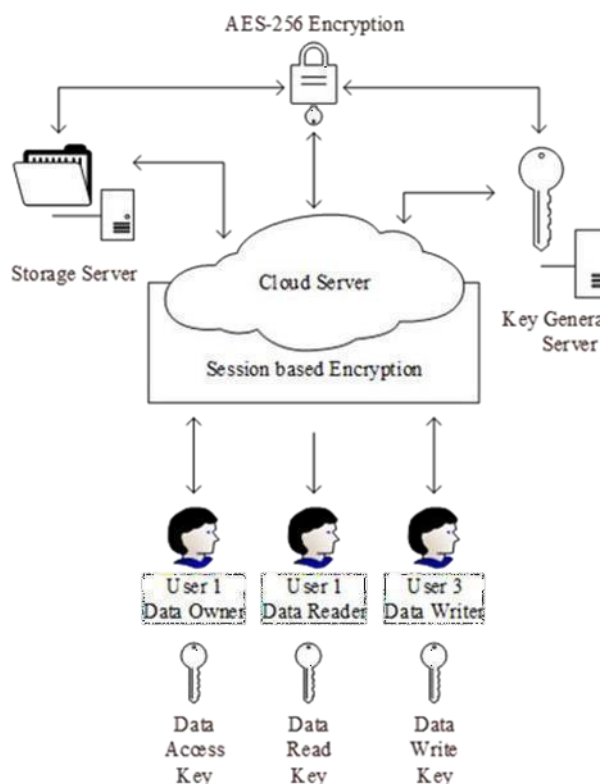
MULTI USER DATA EXCHANGE

Enabling transient multi-user data exchange involves setting up a system where multiple users can securely share data for a limited time period. This can be achieved through various methods, such as using secure file-sharing platforms, implementing encryption, or utilizing temporary storage solutions.

- End to end encryption
- Share data to authorized users only
- Create expiring links or time-limited access to shared files
- Implementing AES algorithm
- Access log detects unauthorized access to storage

MODULES

- Anonymous ID Assignment
- Encryption / Decryption
- Cloud Storage
- Trusted Third Party



3.1 ANONYMOUS ID ASSIGNMENT

In this Anonymous ID Assignment, a variable and an infinite number of repetitions are used for anonymous assignment, and an id-based data sharing mechanism is constructed on top of a safe aggregate data mining process. More specifically, Sturm's theorem and Newton's identities are used in data mining; algorithm scalability is improved by solving some polynomials distributed over finite fields.

3.1.1 GENERATE AID

The administrator of this module wishes to establish the AID for each unique node. Every user presenting on the network might have their own AID created by the administrator. The user can share his data and maintain the confidentiality of his personal information with the aid of this AID.

Creating an anonymous ID assignment involves generating unique identifiers that do not reveal the identity of the users.

- **Generate Unique IDs:** Use a random or pseudorandom algorithm to generate unique IDs for each user. These IDs should be long enough to minimize the chance of collision.
- **Hashing:** Apply a cryptographic hash function to the user's actual identity (e.g., email address, username) to create a hashed version. This ensures that the original identity cannot be reverse-engineered from the anonymous ID.
- **Assignment:** Assign each user a unique anonymous ID generated. This ID will be used to identify the user within the system or application without revealing their actual identity.
- **Data Handling:** When collecting or storing user data, use only the anonymous IDs to identify users. Avoid storing any PII alongside these IDs to prevent potential data breaches or privacy violations.

3.1.2 ASSIGN AID

Every node can have a unique AID from the admin. The network's nodes will communicate with one another via AID. AID is not allowed to hold any private data. AID contributes to maintaining the privacy of personal information.

3.2 ENCRYPTION/DECRYPTION

The system recommends employing the Advanced Encryption System (AES) in this module to encrypt and decrypt user data. Each cipher encrypts and decrypts data in blocks of 256 bits using cryptographic keys with lengths of 256 bits. (Although Rijndael was intended to support different block sizes and key lengths, AES did not use this feature). Symmetric or secret-key ciphers utilize a single key for both encryption and decryption; thus, both the sender and the recipient need to know it and be using it.

AESPROCESS

Key Expansion: AES accepts keys of 128, 192, or 256 bits. If the provided key is not of the required length, a key expansion algorithm is used to generate a set of round keys from the original key.

INITIAL ROUND

The plaintext is divided into blocks, typically 128 bits (16 bytes) for AES. Each block is treated as a 4x4 matrix of bytes. The initial round of AES involves an Add Round Key operation, where each byte of the block is XORed with the corresponding byte of the round key.

MAIN ROUNDS

AES encryption consists of several main rounds, where each round applies a sequence of transformations to the data. Each main round consists of four transformation steps:

SubBytes: Each byte of the block is substituted with a corresponding byte from the S-box lookup table. This step provides confusion.

ShiftRows: Each row of the block is shifted cyclically by a varying offset. This step provides diffusion.

MixColumns: Each column of the block is transformed using a matrix multiplication operation. This step provides further diffusion.

AddRoundKey: Each byte of the block is XORed with the corresponding byte of the round key.

The number of main rounds (Nr) depends on the key length: 10 rounds for a 128-bit key, 12 rounds for a 192-bit key, and 14 rounds for a 256-bit key.

FINAL ROUND

The final round of AES encryption is similar to the main rounds but lacks the MixColumns step.

The final round consists of SubBytes, ShiftRows, and AddRoundKey operations.

3.3 CLOUD STORAGE

Hybrid cloud storage is used in this model of cloud storage. Combining the best features of the previous two models, the hybrid cloud storage model has at least one private cloud and one public cloud architecture. For instance, a business may keep organized and frequently used data in a private cloud and unstructured and archival material in a public cloud.

3.4 TRUSTED THIRD PARTY

A trusted third party is an unbiased, volunteer organization with advanced capabilities that arbitrates disputes and performs audits of public data on behalf of users. To

remotely store data, a user makes use of online infrastructures, platforms, or software for cloud services, which operate in cooperative, parallel, and distributed modes. When a user accesses cloud data, they can communicate with the cloud server independently and independently of others. They are also given complete control over their own data fields. It is crucial to guarantee that data that has been outsourced by users cannot be accessed by unauthorized parties. Furthermore, it is critical to safeguard private information during data access difficulties.

3.5 ALGORITHM: SESSION BASED ENCRYPTION

Session-based encryption refers to the practice of encrypting data during a specific session or interaction between two parties, such as during an online conversation, file transfer, or any data exchange. It typically involves the use of cryptographic techniques to secure the data transmitted between the communicating entities for the duration of that session.

3.5.1 KEY EXCHANGE

At the beginning of a communication session, the entities involved need to establish a secure way to exchange cryptographic keys. This process is often facilitated by protocols like TLS (Transport Layer Security) for web applications. During the key exchange, a shared secret key is generated or exchanged securely between the client and the server.

3.5.2 ENCRYPTION AND DECRYPTION

Once the shared secret key is established, it is used for encrypting and decrypting the data exchanged during the session. Common encryption algorithms like AES (Advanced Encryption Standard) are often employed for this purpose. The data is encrypted by the sender using the shared key and decrypted by the receiver using the same key.

3.5.3 SESSION MANAGEMENT:

The session key is typically used only for the duration of that specific session. Once the session is completed or terminated, the key is no longer valid. This helps in maintaining the security of the communication, as even if someone intercepts the key, it will not be useful for decrypting data in future sessions.

3.5.4 PERFECT FORWARD SECRECY

In some cases, for added security, systems may implement Perfect Forward Secrecy (PFS). PFS ensures that even if a long-term secret key is compromised, it cannot be used to decrypt past communications. New session keys are generated for each session, ensuring that the compromise of one session's key doesn't affect the security of others.

3.5.5 KEY GENERATION

Generating keys typically involves using cryptographic algorithms to create a pair of cryptographic keys: a public key and a private key. These keys are used for various security purposes, including encryption, digital signatures, and secure communication.

IV. RESULT

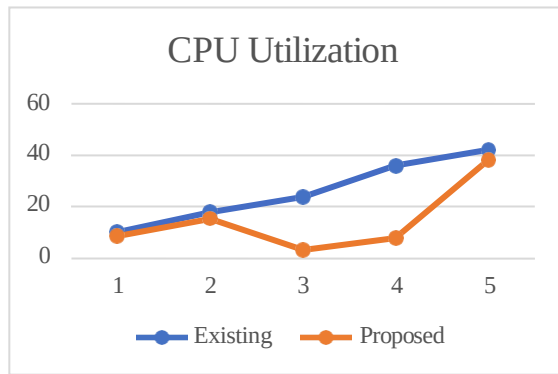


CHART 1: CPU UTILIZATION

Testing Tool: Geekbench provides a cross-platform benchmark that measures CPU and memory performance.

V. CONCLUSION

This work aims to uncover a new privacy issue during cloud computing data access in order to build privacy-preserving access authority sharing. Data confidentiality and integrity are ensured by the establishment of authentication. The exchange of wrapped values during transmission ensures data confidentiality. Anonymous access requests allow users to discreetly communicate their access preferences to the cloud server, enhancing their privacy. Session identifications avoid session correlation by implementing forward security. It suggests using the suggested plan to improve privacy preservation in cloud apps may be possible.

REFERENCES

- [1] C. Castelluccia, A. Chan, E. Mykletun, and G. Tsudik, "Efficient and Provably secure aggregation of encrypted data in wireless sensor networks", Transactions on sensor Networks(TOSN), 2009
- [2] R. Sheikh, B. Kumar, D. Mishra, " Privacy preserving ksecure Sum Protocol", Arxiv Preprint arXiv:0912.0956, 2009
- [3] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu, " Tools for privacy preserving distributed data mining", SIGKDD Explorations Newsletter, 2002
- [4] T. Jung, X. Mao, X.-y. Li, S.-J. Tang, W. Gong, and L. Zhang, " Privacy-preserving data aggregation without secure channel: multivariate polynomial evaluation," in INFOCOM. IEEE, 2013.
- [5] M. Joye and B. Libert, "A scalable scheme for privacy-preserving aggregation of time-series data", in Financial Cryptography and Data Security, (IFCA) 2013.
- [6] W. Dong, V. Dave, L. Qiu, and Y. Zhang, "Secure friend discovery in mobile social networks," in IEEE INFOCOM, 2011, pp. 1647-1655.
- [7] L. Zhang, X. Li, Y. Liu, and T. Jung, " Verifiable private multi-party computation: ranging and ranking", in INFOCOM Mini-conference, IEEE, 2013
- [8] E. Shi, T. Chan, E. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data", in NDSS, vol. 17, 2011.
- [9] M. Ali, R. Dhamotharan, E. Khan, S. Khan, A. Vasilakos, K. Li, and A. Zomaya, "Sedasc: Secure data sharing in clouds," IEEE Systems Journal, vol. 11, no. 2, pp. 395–404, 2017.
- [10] S. Gordon, X. Huang, A. Miyaji, C. Su, K. Sumongkayothin, and K. Wipusitwarakun, "Recursive matrix oblivious ram: An oram construction for constrained storage devices," IEEE Transactions on Information Forensics and Security, vol. 12, no. 12, pp. 3024–3038, 2017.
- [11] X. Chen, X. Huang, J. Li, J. Ma, W. Lou, and D. S. Wong, "New algorithms for secure outsourcing of largescale systems of linear equations," IEEE transactions on information forensics and security, vol. 10, no. 1, pp. 69–78, 2014.
- [12] H. Yuan, X. Chen, J. Li, T. Jiang, J. Wang, and R. Deng, "Secure cloud data deduplication with efficient reencryption," IEEE Transactions on Services Computing, 2019.
- [13] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," Information Sciences, vol. 494, pp. 193–207, 2019.
- [14] S. Challa, A. K. Das, V. Odelu, N. Kumar, S. Kumari, M. K. Khan, and A. V. Vasilakos, "An efficient ecbbased provably secure three-factor user authentication and key agreement protocol for wireless healthcare sensor networks," Computers & Electrical Engineering, vol. 69, pp. 534–554, 2018.

Letter of Acceptance

Details of accepted paper:

Paper ID	Title	Author(s)
ICOECA-071	ENHANCED BLOCKCHAIN BASED DECENTRALIZED PUBLIC AUDITING FOR CLOUD STORAGE	S.L.Jany Shabu, Delphia.J, Preethi A, D.Poornima, K.Karunya, Gayathri S

Herewith, the conference committee of the 4th International Conference on Expert Clouds and Applications (ICOECA 2024) extends its warmest congratulations to you. We are pleased to inform you that after a rigorous peer review process, your aforementioned research paper has been accepted for presentation at the conference. Additionally, your paper has been recommended for inclusion in the ICOECA 2024 conference proceedings.

The ICOECA conference is scheduled to take place on 18-19, April 2024, at RV College of Engineering, Bengaluru, India. We encourage the active participation of highly qualified delegates like yourself to foster the exchange of innovative research ideas and insights.

We congratulate you on being successfully selected for the presentation of your research work in our esteemed conference.


Prof. Deepika K
ICOECA 2024

