

# Cloud Larder is an Increasingly Prevalent Application of Cloud Computing Provide on-demand Outsourcing Document Assistance

Mr. Kandunuri Ramakrishna & Mr. Molkar Rajkumar

Ph.D Research Scholar, Sri Satya Sai University of Technology & Medical Sciences, (SSSUTMS), Sehore, Madhya Pradesh, India. Assistant Professor, KG Reddy College of Engineering and Technology, Hyderabad, India

Assistant Professor<sup>2</sup>, Department of Computer Science Engineering KG Reddy College of Engineering and Technology, Hyderabad, India

**Abstract** - *In this paper, Cloud larder is an increasingly prevalent application of cloud computing, which can provide on-demand outsourcing document assistance for both organizations and individuals. However, users may not fully trust the cloud service providers in that it is difficult to determine whether the Communicating Sequential Processes meet their legal expectations for document security. Therefore, it is critical to develop efficient auditing techniques to strengthen document owners' trust and confidence in cloud storage. a novel public auditing scheme for secure cloud larder based on dynamic hash table, which is a new two-dimensional document structure located at a third parity auditor to record the document property information for dynamic auditing. Differing from the existing works, the proposed scheme migrates the authorized information from the communicating Sequential Processes to the third party auditor, and thereby significantly reduces the computational cost and communication overhead.*

**keywords:** network of nodes, Cloud larder, Batch Auditing, document security, Denial of Service, dynamic auditing, document computation, Error Estimation, test defect data, Software metrics, empirical data..

## I. INTRODUCTION

Cloud larders an important branch of cloud computing, whose goal is to provide powerful and on-demand out-sourcing document assistance for users exploiting highly virtualized infrastructures. Due to the low-cost and high-performance of cloud storage, a growing number of organizations and individuals are tending to outsource their document larder to professional cloud assistance providers, which buoys the rapid development of cloud larder and its relative techniques in recent years. However, as a new cutting-edge technology, cloud larders till faces many security challenges. One of the biggest concerns is how to determine whether a cloud larder system and its provider meet the legal expectations of customers for document security. This is mainly caused by the following reasons. First, cloud users (document owners), who outsource their document in clouds, can no longer verify the integrity of their document via traditional techniques that are often employed in local larder scenarios. Second, Communicating Sequential Processes, which suffer Byzantine failures occasionally, may choose to conceal the document errors from the document owners for their own self- interest.

What is more severe, Communicating Sequential Processes might neglect to keep or even deliberately delete rarely accessed document that belong to ordinary customers to save larder space. Therefore, it is critical and significant to develop efficient auditing techniques to strengthen document owners' trust and confidence in cloud storage, of which the core is how to effectively check document integrity remotely.

So far, many solutions have been presented to overcome this problem, which can be generally divided into two categories: private auditing and public auditing. Private auditing is the initial model for remote checking of document integrity in which the verification operation is performed directly between document owners and Communicating Sequential Processes with relative low cost. However, it cannot provide convincing auditing results, since the owners and Communicating Sequential Processes often mistrust each other. Moreover, it is not advisable for the users to carry out the audit frequently, since it would substantially increase the overhead that the users may not afford. Thus, Ateniese et al. First presented the public auditing scheme, in which the checking work is customarily done by an authorized third party auditor. Compared with the former, the latter can offer dependable auditing results and significantly reduce users' unnecessary burden by introducing an independent TPA. Thus, it is more rational and practical, and prevalent believed to be the right direction of future development.

Document privacy protection document privacy protection has always been an important topic for cloud storage. In the public auditing, the core of this problem is how to preserve uses' privacy while introducing a TPA. Although exploiting document encryption prior to outsourcing is an approach to mitigate the privacy concern in cloud storage, it cannot prevent document leakage during the verification process. Thus, it is important for the cloud auditing to include a privacy-preserving mechanism independent to document encryption. To enhance the efficiency and enable the scalability of public auditing, the TPA should deal with multiple auditing tasks from various users in a fast and cost-efficient manner, i.e., support the batching auditing. As it is well known that a cloud larder system is not just a document warehouse, the users often need to update the document dynamically motivated by various application requirements. Therefore, it is significant for cloud larder auditing to support document dynamics. For the dynamic document auditing, Erway et al. first presented a dynamic provable document

possession scheme, which extends the original PDP model by introducing a rank-based authenticated skip list.

## II. LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool.

Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

Ref 1. Panda: Public Auditing for Shared Document with Efficient User Revocation in the Cloud

Authors: B. Wang, B. Li and H. Li

With document larder and sharing assistance in the cloud, users can easily modify and share document as a group. To ensure shared document integrity can be verified publicly, users in the group need to compute signatures on all the blocks in shared data. Different blocks in shared document are generally signed by different users due to document modifications performed by different users. For security reasons, once a user is revoked from the group, the blocks which were previously signed by this revoked user must be re-signed by an existing user.

The straight forward method, which allows an existing user to download the corresponding part of shared document and re-sign it during user revocation, is inefficient due to the large size of shared document in the cloud. In this project, we propose a novel public auditing mechanism for the integrity of shared document with efficient user revocation in mind.

By utilizing the idea of proxy re-signatures, we allow the cloud to re-sign blocks on behalf of existing users during user revocation, so that existing users do not need to download and re-sign blocks by themselves. In addition, a public verifier is always able to audit the integrity of shared document without retrieving the entire document from the cloud, even if some part of shared document has been re-signed by the cloud. Moreover, our mechanism is able to support batch auditing by verifying multiple auditing tasks simultaneously. Experimental results show that our mechanism can significantly improve the efficiency of user revocation.

Ref 2. Authorized Public Auditing of Dynamic Big Document Larder on Cloud with Efficient Verifiable Fine-grained Updates

Authors: C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan and K. Ramamohanarao

Cloud computing opens a new era in IT as it can provide various elastic and scalable IT assistance in a pay-as-you-go fashion, where its users can reduce the huge capital investments in their own IT infrastructure. In this philosophy, users of cloud larder assistance no longer physically maintain direct control over their data, which makes document security one of the major concerns of using cloud. Existing research

work already allows document integrity to be verified without possession of the actual document file. When the verification is done by a trusted third party, this verification process is also called document auditing, and this third party is called an auditor. However, such schemes in existence suffer from several common drawbacks.

First, a necessary authorization/authentication process is missing between the auditor and cloud service provider, i.e., anyone can challenge the cloud service provider for a proof of integrity of certain file, which potentially puts the quality of the so-called 'auditing-as-a-service' at risk. Second, although some of the recent work based on BLS signature can already support fully dynamic document updates over fixed-size document blocks, they only support updates with fixed-sized blocks as basic unit, which we call coarse-grained updates.

As a result, every small update will cause re-computation and updating of the authenticator for an entire file block, which in turn causes higher larder and communication overheads. In this project, we provide a formal analysis for possible types of fine-grained document updates and propose a scheme that can fully support authorized auditing and fine-grained update requests. Based on our scheme, we also propose an enhancement that can dramatically reduce communication overheads for verifying small updates. Theoretical analysis and experimental results demonstrate that our scheme can offer not only enhanced security and flexibility. Cloud Security Auditing: Challenges and Emerging Approaches

Authors: J. Ryoo, S. Rizvi, W. Aiken and J. Kissell

IT auditors collect information on an organization's information systems, practices, and operations and critically analyze the information for improvement. One of the primary goals of an IT audit is to determine if the information system and its maintainers are meeting both the legal expectations of protecting customer document and the company standards of achieving financial success against various security threats. These goals are still relevant in the newly emerging cloud computing model of business, but they need customization. There are clear differences between cloud and traditional IT security auditing. In this article, the authors explore potential challenges unique to cloud security auditing; examine additional challenges specific to particular cloud computing domains such as banking, medical, and government sectors; and present emerging cloud-specific security auditing approaches and provide critical analysis.

Ref 4. Privacy-Preserving Public Auditing for Secure Cloud Storage

Authors: C. Wang, S. M. Chow, Q. Wang, K. Ren and W. Lou

Using cloud storage, users can remotely store their document and enjoy the on-demand high-quality applications and assistance from a shared pool of configurable computing resources, without the burden of local document larder and maintenance. However, the fact that users no longer have physical possession of the outsourced document makes the document integrity protection in cloud computing a formidable task, especially for users with constrained computing resources.

Moreover, users should be able to just use the cloud larder as if it is local, without worrying about the need to verify its integrity. Thus, enabling public audit ability for cloud larder is of critical importance so that users can resort to a third-party auditor (TPA) to check the integrity of outsourced document and be worry free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities toward user document privacy, and introduce no additional online burden to user. In this project, we propose a secure cloud larder system supporting privacy-preserving public auditing.

We further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient.

### III. SYSTEM OVERVIEW

#### Proposed System:

We present a novel public auditing scheme for secure cloud larder based on dynamic hash table (DHT), which is a new two-dimensional document structure located at a third parity auditor to record the document property information for dynamic auditing. Differing from the existing works, the proposed scheme migrates the authorized information from the CSP to the TPA, and thereby significantly reduces the computational cost and communication overhead.

#### Advantages

Privacy preservation by combining the homomorphic authenticator based on the public key with the random masking generated by the TPA, and achieve batch auditing by employing the aggregate BLS signature technique.

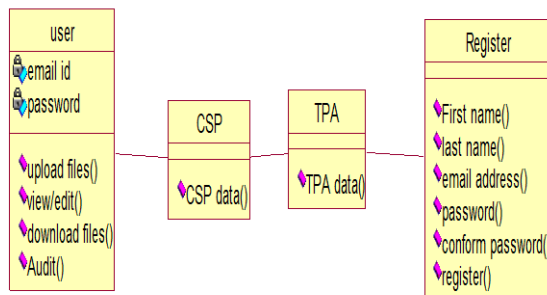
The security of the proposed scheme, and evaluate the auditing performance by detailed experiments and comparisons with the existing ones.

We proved that the satisfiability of a node  $v$  . the given semantics can be determined by  $v$ 's child nodes.

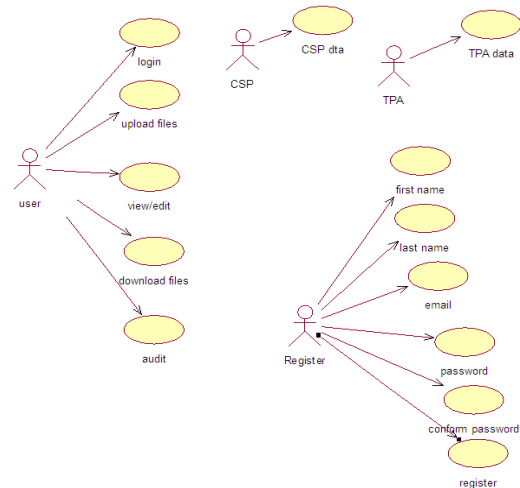
Another salient feature is that our approach is independent of query semantics

#### Design: Class Diagram:

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



A USE CASE DIAGRAM IN THE UNIFIED MODELING LANGUAGE (UML) IS A TYPE OF BEHAVIORAL DIAGRAM DEFINED BY AND CREATED FROM A USE-CASE ANALYSIS. ITS PURPOSE IS TO PRESENT A GRAPHICAL OVERVIEW OF THE FUNCTIONALITY PROVIDED BY A SYSTEM IN TERMS OF ACTORS, THEIR GOALS (REPRESENTED AS USE CASES), AND ANY



dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use case and actors.

An actor is represents a user or another system that will interact with the system you are modeling. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

### IV. SYSTEM IMPLEMENTATION

In this architecture, we concentrate on the design of an effective public auditing scheme based on the DHT illustrated in figure, which involves the following three entities: User:

Who stores a great quantity of document files in the cloud, can be an individual or a organization.

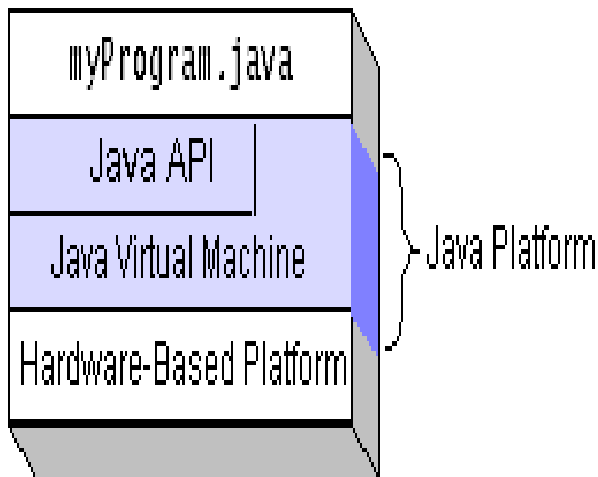
Cloud Service Provider (CSP):

Who manages and coordinates a number of cloud servers to offer scalable and on-demand outsourcing document assistance for users.

Third Party Auditor (TPA):

Who can verify the reliability of the cloud larder assistance credibly and dependably on behalf of the users upon request? Users can be relieved of the burden of larder and computation while enjoying the larder and maintenance service by outsourcing their document into the CSP. However, due to the loss of local possession of the data, they are keen to

ensure the correctness and integrity of their document periodically. To obtain a convincing answer as well as alleviate the users' burden potentially induced by the frequent verification, the TPA is involved to check the integrity of the users' document stored in the cloud. However, in the whole verification process, the TPA is not expected to be able to learn the actual content of the users' document for privacy protection.



Modules :There are three modules

Privacy- preserving  
Batch auditing  
Dynamic auditing  
Privacy-preserving:

Document privacy protection (DPP) has always been an important topic for cloud storage. In the public auditing, the core of this problem is how to preserve users' privacy while introducing a TPA. Although exploiting document encryption prior to outsourcing is an approach to mitigate the privacy concern in cloud larder, it cannot prevent document leakage during the verification process. Thus, it is important for the cloud auditing to include a privacy-preserving mechanism independent to document encryption.

Batch auditing:

To enhance the efficiency and enable the scalability of public auditing, the TPA should deal with multiple auditing tasks from various users in a fast and cost- efficient manner, i.e., support the Batching auditing.

Dynamic auditing:

As it is well known that a cloud larder system is not just a document warehouse, the users often need to update the document dynamically motivated by various application requirements. Therefore, it is significant for cloud larder auditing to support document dynamics.

## V. SOFTWARE DESCRIPTION

### JAVA TECHNOLOGY

Java technology is both a programming language and a

platform.

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

One can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most prevalent platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The Java Virtual Machine (Java VM)
- The Java Application Programming Interface (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code.

What Can Java Technology Do?



The most common types of programs written in the Java programming language are applets and applications. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java servlets are a prevalent choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

#### SAMPLE CODE

//DHTServlet.java

```
package org.project;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.mail.Session;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
```

```
public class DHTServlet extends HttpServlet{
    private static final long serialVersionUID = 1L;
    DBProcess dbProcess;
    public void doGet(HttpServletRequest request,
        HttpServletResponse response){
        String query = request.getParameter(query);
        if(user.equals(query)){
            try{
                HttpSession session = request.getSession(false);
                User user = (User) session.getAttribute(user);
                if(user==null user.getEmail().isEmpty()){
                    RequestDispatcher view =
                        request.getRequestDispatcher(jspuserLogin.jsp);
                    view.forward(request, response);
                }
            } else{
                RequestDispatcher view =
                    request.getRequestDispatcher(jspuserHome.jsp);
                view.forward(request, response);
            } }
            catch(Exception ex){
                ex.printStackTrace();
            } }
            else if(userUpload.equals(query)){
                try{
                    HttpSession session = request.getSession(false);
                    User user = (User) session.getAttribute(user);
                    if(user==null user.getEmail().isEmpty()){
                        RequestDispatcher view =
                            request.getRequestDispatcher(jspuserLogin.jsp);
                            view.forward(request, response);
                        }
                    } else{
                        RequestDispatcher view =
                            request.getRequestDispatcher(jspuserUpload.jsp);
                            view.forward(request, response);
                        } }
                    catch(Exception ex){
                        ex.printStackTrace();
                    } }
                    else if(userViewAll.equals(query)){
                        try{
                            HttpSession session = request.getSession(false);
                            User user = (User) session.getAttribute(user);
                            if(user==null user.getEmail().isEmpty()){
                                RequestDispatcher view=
                                    request.getRequestDispatcher(jspuserLogin.jsp);
                                    view.forward(request, response);
                                }
                            } else{
                                ListFileBlock fileList = new ArrayList();
                                fileList = dbProcess.getAllFilesForUser(user.getUserId());
                                request.setAttribute(fileList, fileList);
                                RequestDispatcher view=
                                    request.getRequestDispatcher(jspuserView.jsp);
                                    view.forward(request, response);
                                }
                            }
                        }
                    }
```

## VI. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS: Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

### Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

### Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; document fields, predefined processes, and successive processes must be considered for

testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

### White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level. White box testing also known as glass box testing, transparent box testing, and structural testing is a method of testing software that tests internal structures or workings of an application.

### Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black Box Testing also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is testing in which the software under test is treated, as a black box. The test provides inputs and responds to outputs without considering how the software works.

### Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

### Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

### Test objectives

All field entries must work properly.

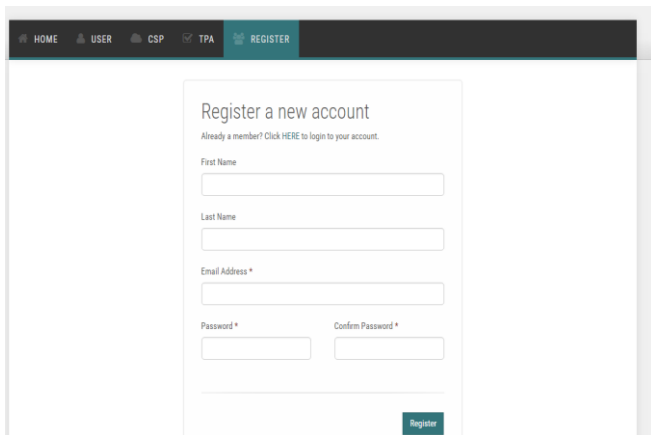
Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

## VII. RESULTS

Dynamic-hash-table based public auditing for secure cloud ladders prevalent to introduce an authenticated document structure to achieve dynamic auditing. Cloud ladder auditing has attracted increasing attention. In earliest days, they introduce index-hash-table for public auditing. Although their scheme cannot support public auditing, they reveal a general approach to achieve dynamic auditing.

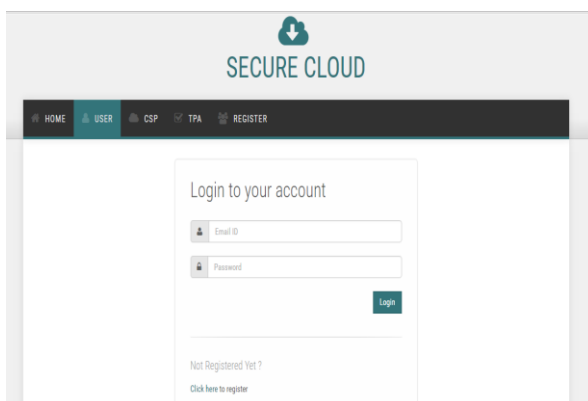


The screenshot shows a web application interface for registration. At the top, there is a navigation bar with links: HOME, USER, CSP, TPA, and REGISTER. The main content area is titled 'Register a new account' and includes a link for existing members to log in. The registration form consists of several input fields: 'First Name', 'Last Name', 'Email Address \*', 'Password \*', and 'Confirm Password \*'. A green 'Register' button is located at the bottom right of the form.

#### Registration Page:

User who want to store information in the cloud, they must be registered.

Now we present a novel public auditing scheme for secure cloud larder based on dynamic hash table , which is a new two-dimensional document structure located at a third parity auditor to record the document property information for dynamic auditing. Differing from the existing works. The proposed scheme migrates the authorized information from the CSP to the TPA, and thereby significantly reduces the computational cost and communication overhead. Privacy preservation by combining the homomorphic authenticator based on the public key with the random masking generated by the TPA, and achieves batch auditing by employing the aggregate BLS signature technique. The security of the proposed scheme, and evaluate the auditing performance by detailed experiment. Another salient feature is that our approach is independent of query semantics.



The screenshot shows a web application interface for login. At the top, there is a navigation bar with links: HOME, USER, CSP, TPA, and REGISTER. The main content area is titled 'Login to your account' and includes a link for new users to register. The login form consists of two input fields: 'Email ID' and 'Password'. A green 'Login' button is located at the bottom right of the form.

Fig : Login Page

### VIII. CONCLUSION AND FUTURE WORK

Nowadays, cloud larder can offer on-demand outsourcing document assistance for both organizations and individuals has

been attracting more and more attention. However, one of the most serious obstacles to its development is that users may not fully trust the Communicating Sequential Processes in that it is difficult to determine whether the Communicating Sequential Processes meet their legal expectations for document security. Therefore, it is critical and significant to develop efficient auditing techniques to strengthen document owners' trust and confidence in cloud storage. In this project, we are motivated to present a novel public auditing scheme for secure cloud larder using dynamic hash table (DHT), which is a new two dimensional document structure used to record the document property information for dynamic auditing. Differing from the existing works, our scheme migrates the auditing met document excerpt the block tags from the CSP to the TPA, and thereby significantly reduces the computational cost and communication overhead. Meanwhile, exploiting the structural advantages of the DHT, our scheme can also achieve better performance than the state-of-the-art schemes in the updating phase. In addition, for privacy preservation, our scheme introduces a random masking provided by the TPA into the process of generating proof to blind the document information. Moreover, our scheme further exploits the aggregate BLS signature technique from bilinear maps to perform multiple auditing tasks simultaneously, of which the principle is to aggregate all the signatures by different users on various document blocks into a single short one and verify it for only one time to reduce the communication cost in the verification process. We formally prove the security of our scheme, and evaluate the auditing performance by detailed experiments and comparisons with the existing ones. The results demonstrate that our scheme can effectively achieve secure auditing in clouds, and induce significantly fewer costs of storage, communication and computation than the previous schemes.

#### REFERENCES

- [1].Dewan and R. C. Hansdah. "A Survey of Cloud LarderFacilities ", Proc. 7th IEEE World Congress on Assistance, pp. 224- 231, July 2011.
- [2].C. Wang, Q. Wang, K. Ren, N. Cao and W. Lou. "Toward Secure and Dependable LarderAssistance in Cloud Computing", IEEE Trans. Service Computing, vol. 5, no. 2, pp. 220- 232, 2012.
- [3].K. Ren, C. Wang and Q. Wang. "Security Challenges for the Public Cloud," IEEE Internet Computing, vol. 16, no. 1, pp. 69– 73, 2012.
- [4].J. Ryoo, S. Rizvi, W. Aiken and J. Kissell. "Cloud Security Auditing: Challenges and Emerging Approaches", IEEE Security & Privacy, vol. 12, no. 6, pp. 68- 74, 2014.
- [5].C. Wang, K. Ren, W. Lou and J. Li. "Toward Publicly Auditable Secure Cloud Document LarderAssistance", IEEE network, vol. 24, no. 4, pp. 19- 24, 2010.
- [6].Q. Wang, C. Wang, K. Ren, W. Lou and J. Li. "Enabling Public Auditability and Document Dynamics for LarderSecurity in Cloud Computing," IEEE Trans. on Parallel and Distributed Systems, vol. 22, no. 5, pp. 847- 859, 2011.

- [7]. F. Sebé, J. Domingo- Ferrer, A. Martínez- Ballesté, Y. Deswarte and J.- J. Quisquater, "Efficient Remote Document Possession Checking in Critical Information Infrastructures," IEEE Trans. Knowledge Document Eng., vol. 20, no. 8, pp. 1034- 1038, 2008.
- [8]. G. Ateniese, R.B. Johns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable Document Possession at Untrusted Stores," Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS), pp. 598- 609, 2007.
- [9]. K. Yang and X. Jia. "Document Larder Auditing Service in Cloud Computing: Challenges, Methods and Opportunities". World Wide Web, vol. 15, no. 4, pp. 409- 428, 2012
- [10]. C. Wang, Q. Wang, K. Ren and W. Lou, "Privacy- Preserving Public Auditing for Document Larder Security in Cloud Computing," Proc. IEEE INFOCOM, pp. 1- 9, 2010.
- [11]. C. Wang, S. M. Chow, Q. Wang, K. Ren and W. Lou, "Privacy- Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. on Computers, vol. 62, no. 2, pp. 362- 375, 2013.
- [12]. Y. Zhu, H. Hu, G. Ahn, and M. Yu, "Cooperative Provable Document Possession for Integrity Verification in Multi- Cloud Storage," IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 12, pp. 2231- 2244, 2012.
- [13]. K. Yang and X. Jia, "An Efficient and Secure Dynamic Auditing Protocol for Document Larder in Cloud Computing," IEEE Trans. On Parallel and Distributed Systems, vol. 24, no. 9, pp. 1717- 1726, 2013.
- [14]. C. C. Erway, A. Küpçü, C. Papamanthou and R. Tamassia. "Dynamic Provable Document Possession," Proc. 16th ACM Conf. Computer and Comm. Security, pp. 213- 222, 2009.
- [15]. Y. Zhu, H. Wang, Z. Hu, G.- J. Ahn, H. Hu and S. S. Yau, "Dynamic Audit Assistance for Outsourced Larder in Clouds," IEEE Trans. on Assistance Computing, vol. 6, no. 2, pp. 227-238, 2013.
- [16]. D. Boneh, B. Lynn and H. Shacham, "Short Signatures from the Weil Pairing," Proc. ASIACRYPT, vol. 2248, LNCS, pp. 514- 532, 2001.
- [17]. B. Wang, B. Li and H. Li. "Panda: Public Auditing for Shared Document with Efficient User Revocation in the Cloud", IEEE Trans. on Service Computing, vol. 8, no. 1, pp. 92- 106, 2015.
- [18]. C. Liu, R. Ranjan, X. Zhang, C. Yang, D. Georgakopoulos and J. Chen. "Public Auditing for Big Document Larder in Cloud Computing- - A Survey", Proc. 16th IEEE International Conf. Computational Science and Engineering (CSE), pp. 1128- 1135, 2013.

#### Biography:



Mr. K. Ramakrishna, (Ph.D) Research Scholar in Sri Satya Sai University of Technology and Medical Sciences, Sehare (M.P) India. Working as a Assistant Professor in computer science and engineering department in KG Reddy college of engineering and technology, Hyderabad, India. He received the master of technology degree in VNR Vignana Jyothi Institute of Engineering and Technology- Jawaharlal Nehru Technological University Hyderabad, India. He received the bachelor of technology degree in The Vazir Sultan College of Engineering and technology, kakatiya university, Warangal, India. His Research Interests Include mobile ad-hoc networks, Document Mining, Information Security, Software Testing, Soft computing, Software Engineering, mobile communication and cloud computing.



Mr. Molkar Rajkumar, assistant professor, Department of Computer Science Engineering , KG Reddy College of Engineering and Technology, Hyderabad, India. He Has 8+ Years Teaching Experience. His Research Interests Include cloud computing, Data Mining, Information Security, Software Testing, mobile communication and mobile ad-hoc networks.