

Enabling Ternary Hash Tree based Integrity Verification for Secure Cloud Data Storage

A PROJECT REPORT

Submitted by Monishraj.M

Reg No:38290108

&

Sabarinath.P

Reg No:38290075

III YEAR BSc CSE

Sathyabama Institute Of Science And Technology

Guided By: KEERTHI SAMHITHA.

BONAFIDE CERTIFICATE

ABSTRACT

Cloud Computing enables the remote users to access data, services, and applications in on-demand from the shared pool of configurable computing resources, without the consideration of storage, hardware and software management. On the other hand, it is not easy for cloud users to identify whether Cloud Service Provider's (CSP) tag along with the data security legal

expectations. So, cloud users could not rely on CSP's in terms of trust. So, it is significant to build a secure and efficient data auditing framework for increasing and maintaining cloud users trust with CSP. Researchers suggested introducing Third Party Auditor (TPA) on behalf of cloud user for verifying the outsourced data integrity, which may reduce the computation overhead of cloud users. In this work, we proposed a novel integrity verification framework for securing cloud storage based on Ternary Hash Tree (THT) and Replica based Ternary Hash Tree (R-THT), which will be used by TPA to perform data auditing. Differing from existing work, the proposed framework performs Block-level, File-level and Replica-level auditing with tree block ordering, storage block ordering for verifying the data integrity and ensuring data availability in the cloud. We further extend our framework to support error localization with data correctness, dynamic updates with block update, insert and delete operations in the cloud. The structure of THT and R-THT will reduce the computation cost and provide efficiency in data updates compared to the existing schemes. The security analysis of the proposed public auditing framework indicates the achievement of desired properties and performance has been evaluated with the detailed experiment set. The results show that the proposed secure cloud auditing framework is highly secure and efficient in storage, communication and computation costs.

TABLE OF CONTENTS

CHAPTER No	TITLE	PAGE
	LIST OF FIGURES	Vi

	LIST OF ABBREVIATIONS	Vii
1	INTRODUCTION	
1.1	About the Project	
2	SYSTEM ANALYSIS	
2.1	Existing system	
2.2	Proposed system	
3	REQUIREMENTS SPECIFICATION	
3.1	Introduction	
3.2	Hardware and Software specification	
3.3	Technologies Used	
3.3.1	Java	
3.3.3.1	Introduction to Java	
3.3.1.2	Working of Java	
4	Project Purpose and Scope	
4.1	Purpose	
4.2	Project Scope	
4.3	Product Perspective	
4.4	System Features	

4.5	Design and Implementation Constraints	
4.6	Other Nonfunctional Requirements	
5	SYSTEM DESIGN	
5.1	Architecture Diagram	
6	SYSTEM DESIGN – DETAILED	
6.1	Modules	
6.2	Module explanation	
7	CODING AND TESTING	
7.1	Coding	
7.2	Coding standards	
7.3	Test procedure	
7.4	Test data and output	
	SOURCE CODE	
	SNAP SHOTS	
	REFERENCES	

LIST OF FIGURES

5.1 System Design

5.2 Sequence Diagram

5.3 Use Case Diagram

5.4 Activity Diagram

5.5 Collaboration Diagram:

5.6 Data Flow Diagram

5.7 Class Diagram

LIST OF ABBREVIATIONS

JDK	Java Development Toolkit
DEX	Dalvik Executables
TCP	Transmission Control Protocol
IP	Internet Protocol
HTTP	Hyper Text Transfer Protocol
ADT	Android Development Tool

CHAPTER 1

INTRODUCTION

Aim:

The main aim of this project is to provide a reliable and secure cloud service and also increase trustworthiness of certifications by Continuous Automatic Auditing.

Synopsis:

Cloud Computing enables the remote users to access data, services, and applications in on-demand from the shared pool of configurable computing resources, without the consideration of storage, hardware and software management. On the other hand, it is not easy for cloud users to identify whether Cloud Service Provider's (CSP) tag along with the data security legal expectations. So, cloud users could not rely on CSP's in terms of trust. So, it is significant to build a secure and efficient data auditing framework for increasing and maintaining cloud users trust with CSP. Researchers suggested introducing Third Party Auditor (TPA) on behalf of cloud user for verifying the outsourced data integrity, which may reduce the computation overhead of cloud users. In this work, we proposed a novel integrity verification framework for securing cloud storage based on Ternary Hash Tree (THT) and Replica based Ternary Hash Tree (R-THT), which will be used by TPA to perform data auditing. Differing from existing work, the proposed framework performs Block-level, File-level and Replica-level auditing with tree block ordering, storage block ordering for verifying the data integrity and ensuring data availability in the cloud. We further extend our framework to support error localization with data correctness, dynamic updates with block update, insert and delete operations in the cloud. The structure of THT and R-THT will reduce the computation cost and provide efficiency in data updates compared to the existing schemes. The security analysis of the proposed public auditing framework indicates the achievement of desired properties and performance has been evaluated

with the detailed experiment set. The results show that the proposed secure cloud auditing framework is highly secure and efficient in storage, communication and computation costs.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In cloud computing, remote data integrity checking is an important security problem. The client's massive data is outside his control. The malicious cloud server may corrupt the client's data in order to gain more benefits. However, cloud services are part of an ever changing environment, resulting from fast technology life cycles and inherent cloud computing (CC) characteristics, like on-demand provisioning and entangled supply chains. Hence, such long validity periods may put in doubt reliability of issued certifications. And also cloud service customers do not longer possess their data locally, assuring that their data is being correctly stored and integrity is maintained in cloud environments is of critical importance. Data integrity may be threatened by, for example, malicious insiders, data loss, technical failures, and by external attackers.

Problem Definition

1. Long validity periods may put in doubt reliability of issued certifications.
2. Data Integrity.
3. Procrastinating third-party auditors.

2.2 PROPOSED SYSTEM

In MultiCloud environment, remote data integrity checking is required to secure user's data. User will upload file to Cloud. This file is split into blocks using Dynamic Block generation Algorithm. The Blocks are stored in Ternary Hash Tree (THT) format. The blocks have a parents node and child node. File Allocation Table (FAT) File System has proper Indexing and Metadata's for the different Chunks of the Cloud Storage. Here the auditor agrees to inspect logs,

which are routinely created during monitoring operations by services providers to assess certification adherence. If Attacker corrupts data in MultiCloud, the continuous auditing process helps the verifier to perform Block and File level checking for remote data Integrity Checking using Verifiable Data Integrity Checking Algorithm. The auditing processes have a flow, first the parent block checking. If the parent block have any corrupted file then the child node auditing. If the child nodes have any corrupted file the File recovery is done by the Verifier automatically if the data gets corrupted during checking. Users can complaint cloud for file recovery.

Advantages:

- The proposed framework performs Block-level, File-level and Replica-level auditing with tree block ordering.
- Our framework to support error localization with data correctness, dynamic updates with block update, insert and delete operations in the cloud.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 INTRODUCTION

NOWADAYS, it has become a common practice for education, healthcare, and other sectors to utilize the cloud environment for storing and processing the business data, for acting as a backup repository to personal data such as photographs, mail archives, contact details, and so on. Cloud services provide a massive amount of computing resources and storage space, which makes the users eliminate the accountability of maintaining data at local machines. So, users completely depend on the cloud services for their data storage, availability, and integrity [30]. Even though cloud computing infrastructures and service providers are much reliable and powerful, a wide range of threats for data integrity exists. Incidents of data loss and outages of significant cloud services appear day to day]. On the other hand, Cloud Service Provider (CSP) may face Byzantine failure problems occasionally, may remove hardly ever accessed files and may decide not to showcase the data loss errors to DO for maintaining the reputation [1][2][3]. Unless the third party auditor (TPA) or the Data Owner (DO) verifies the integrity of the data, the level of data confidentiality of the outsourced data is unknown. The major concern with the cloud data access is to ensure data integrity at untrusted cloud servers. Many researchers [5-23] have proposed various schemes to address the data integrity verification issues in the cloud. In existing schemes, the role of verifier falls either in private data auditing or in public data auditing. In private data auditing, DO takes the responsibility to verify the correctness of the stored data in CSP. In public data auditing, DO rely on trusted third-party auditor for challenging CSP against data integrity verification. However, data physical possession in the cloud is not known to DO. So, both types of auditing are performed without knowing the contents of actual data files. In the cloud, remotely stored data may be accessed to perform read and write operations. In order to perform a write operation, the data need to be available for dynamic access. But, many existing schemes focus on read access (i.e) static file access and the support of

dynamic data updates is limited to the DO. The inconsistent data blocks are capable of being found through data auditing techniques, but the methodology to localize the error blocks and correcting them across the replica blocks is left as an open issue.

3.2 HARDWARE AND SOFTWARE SPECIFICATION

3.2.1 HARDWARE REQUIREMENTS

- ☐ Hard Disk : 200GB and Above
- ☐ RAM : 2GB and Above
- ☐ Processor : i3 and Above

3.2.2 SOFTWARE REQUIREMENTS

- ☐ Windows XP and Above
- ☐ JDK 1.7
- ☐ Tomcat 6.0
- ☐ Tomcat 7.0
- ☐ MySql 5.0

3.3 TECHNOLOGIES USED

- ☐ J2EE (JSP, Servlet)
- ☐ Struts Framework
- ☐ JavaScript , Ajax , HTML ,CSS, jquery
- ☐ Web Services (JAX -WS), JSON.

3.3.1 JAVA

Java is an object-oriented programming language developed initially by James Gosling and colleagues at Sun Microsystems. The language, initially called Oak (named after the oak trees outside Gosling's office), was intended to replace C++, although the feature set better resembles that of Objective C.

3.3.1.1 INTRODUCTION TO JAVA

Java has been around since 1991, developed by a small team of Sun Microsystems developers in a project originally called the Green project. The intent of the project was to develop a platform-independent software technology that would be used in the consumer electronics industry. The language that the team created was originally called Oak.

The first implementation of Oak was in a PDA-type device called Star Seven (*7) that consisted of the Oak language, an operating system called GreenOS, a user interface, and hardware. The name *7 was derived from the telephone sequence that was used in the team's office and that was dialed in order to answer any ringing telephone from any other phone in the office.

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical

Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called Java. Soon after, the Java compiler was rewritten in the Java language from its original C code, thus proving that Java could be used effectively as an application language. Sun introduced Java in May 1995 at the Sun World 95 convention.

Web surfing has become an enormously popular practice among millions of computer users. Until Java, however, the content of information on the Internet has been a bland series of HTML documents. Web users are hungry for applications that are interactive, that users can execute no matter what hardware or software platform they are using, and that travel across heterogeneous networks and do not spread viruses to their computers. Java can create such applications.

3.3.1.2 WORKING OF JAVA

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data (called attributes) and functions (called methods).

When the interpreter executes a class, it looks for a particular method by the name of **main**, which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the `argv []` of C), and is declared as a static method.

To output text from the program, we execute the **println** method of **System.out**, which is java's output stream. UNIX users will appreciate the theory behind such a stream, as it is actually standard output. For those who are instead used to the Wintel platform, it will write the string passed to it to the user's program.

Java consists of two things :

- ☐ Programming language
- ☐ Platform

3.3.1.3 THE JAVA PROGRAMMING LANGUAGE

Java is a high-level programming language that is all of the following:

- ☐ Simple
- ☐ Object-oriented
- ☐ Distributed
- ☐ Interpreted
- ☐ Robust
- ☐ Secure

- ☐ Architecture-neutral
- ☐ Portable
- ☐ High-performance
- ☐ Multithreaded
- ☐ Dynamic

The code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works:

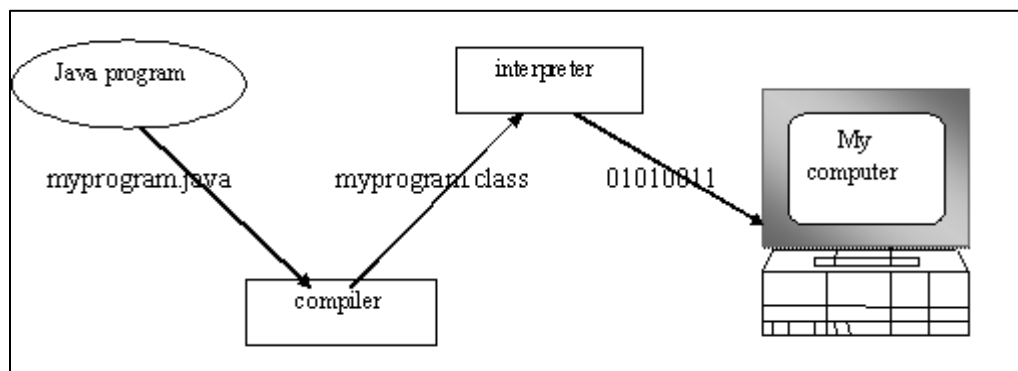
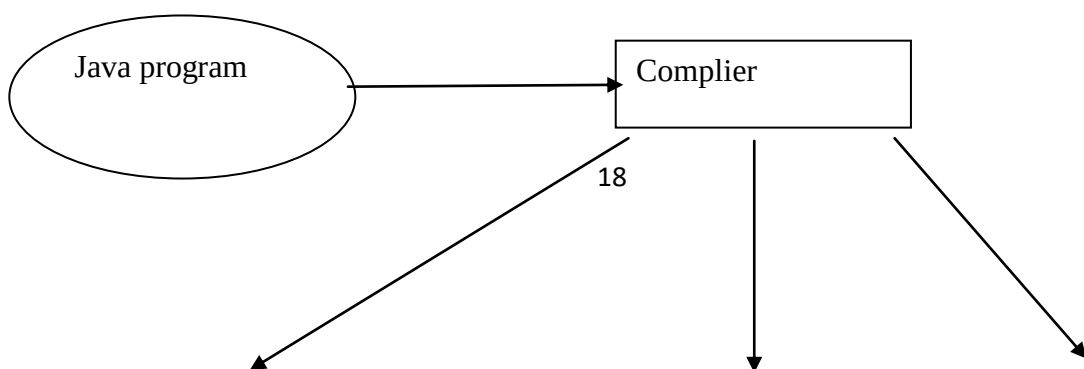


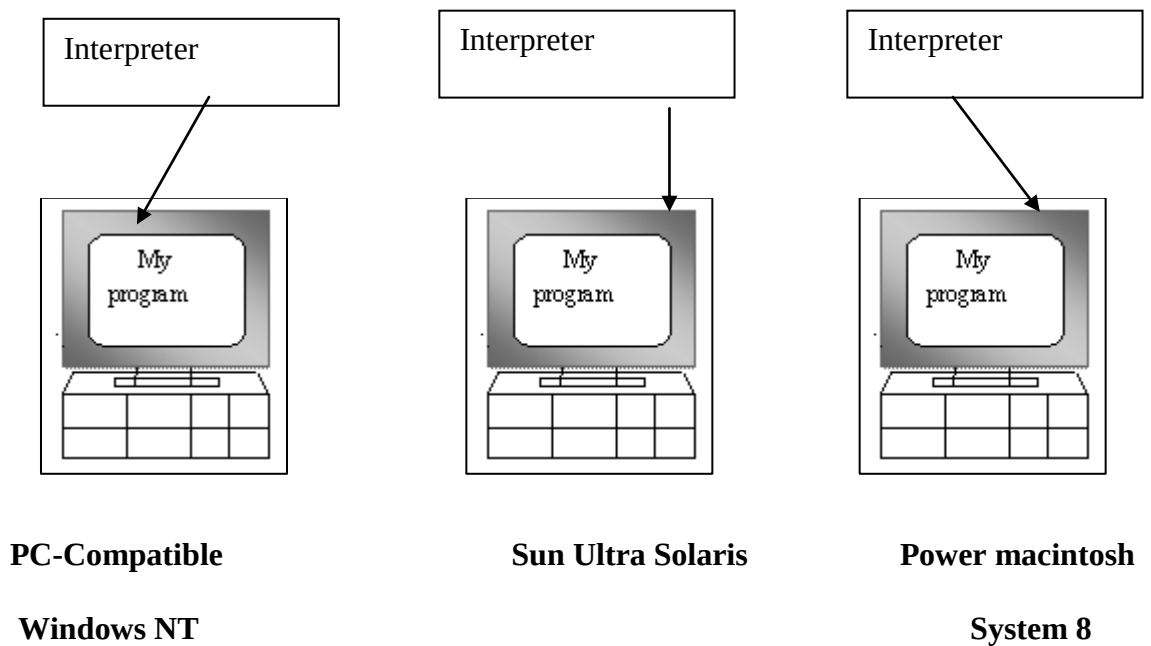
Fig.3.1

You can think of Java byte codes as the machine code instructions for the **Java Virtual Machine (JVM)**. Every Java interpreter, whether it's a Java development tool or a Web browser

that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make “write once, run anywhere” possible.

You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can be run on Windows NT, Solaris and Macintosh





3.3.1.4 THE JAVA PLATFORM

A platform is the hardware or software environment in which a program runs. 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. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components :

- ☐ The Java Virtual Machine (JVM)
- ☐ The Java Application Programming Interface (Java API)

You've already been introduced to the JVM. 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 (**packages**) of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

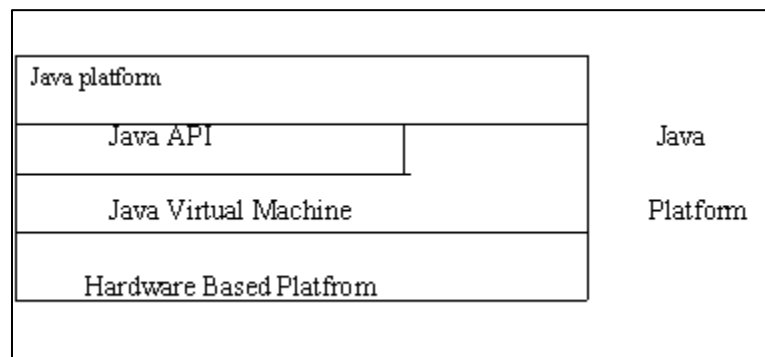


Fig.3.3

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, weel-tuned interpreters, and just-in-time byte compilers can bring Java's performance close to that of native code without threatening portability.

3.3.2 APACHE TOMCAT SERVER

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an

environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

Environment

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

Product features

Tomcat 3.x (initial release)

- implements the Servlet 2.2 and JSP 1.1 specifications
- servlet reloading
- basic HTTP functionality Tomcat 4.x
- implements the Servlet 2.3 and JSP 1.2 specifications

- servlet container redesigned as Catalina
- JSP engine redesigned as Jasper
- Coyote connector
- Java Management Extensions (JMX), JSP and Struts-based administration
- Tomcat 5.x
- implements the Servlet 2.4 and JSP 2.0 specifications
- reduced garbage collection, improved performance and scalability
- native Windows and Unix wrappers for platform integration
- faster JSP parsing

History

Tomcat started off as a servlet specification implementation by James Duncan Davidson, a software architect at Sun. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation.

Davidson had initially hoped that the project would become open-sourced and, since most open-source projects had O'Reilly books associated with them featuring an animal on the cover, he wanted to name the project after an animal. He came up with Tomcat since he reasoned the animal represented something that could take care of and fend for itself. His wish to see an animal cover eventually came true when O'Reilly published their Tomcat book with a tomcat on the cover.

3.3.3 Data Mining Introduction:

In general terms, “**Mining**” is the process of extraction of some valuable material from the earth e.g. coal mining, diamond mining etc. In the context of computer science, “Data

Mining” refers to the extraction of useful information from a bulk of data or data warehouses. One can see that the term itself is a little bit confusing. In case of coal or diamond mining, the result of extraction process is coal or diamond. But in case of Data Mining, the result of extraction process is not data!! Instead, the result of data mining is the patterns and knowledge that we gain at the end of the extraction process. In that sense, Data Mining is also known as Knowledge Discovery or Knowledge Extraction.

Gregory Piatetsky-Shapiro coined the term “Knowledge Discovery in Databases” in 1989. However, the term ‘data mining’ became more popular in the business and press communities. Currently, Data Mining and Knowledge Discovery are used interchangeably.

Now a days, data mining is used in almost all the places where a large amount of data is stored and processed. For example, banks typically use ‘data mining’ to find out their prospective customers who could be interested in credit cards, personal loans or insurances as well. Since banks have the transaction details and detailed profiles of their customers, they analyze all this data and try to find out patterns which help them predict that certain customers could be interested in personal loans etc.

Main purpose of Data mining

Basically, the information gathered from Data Mining helps to predict hidden patterns, future trends and behaviors and allowing businesses to take decisions.

Technically, data mining is the computational process of analyzing data from different perspective, dimensions, angles and categorizing/summarizing it into meaningful information.

Data Mining can be applied to any type of data e.g. Data Warehouses, Transactional Databases, Relational Databases, Multimedia Databases, Spatial Databases, Time-series Databases, World Wide Web.

Datamining as a whole process

The whole process of Data Mining comprises of three main phases:

1. Data Pre-processing – Data cleaning, integration, selection and transformation takes place
2. Data Extraction – Occurrence of exact data mining
3. Data Evaluation and Presentation – Analyzing and presenting results.

Why is data mining important?

So why is data mining important? You've seen the staggering numbers – the volume of data produced is doubling every two years. Unstructured data alone makes up 90 percent of the digital universe. But more information does not necessarily mean more knowledge.

Data mining allows you to:

- Sift through all the chaotic and repetitive noise in your data.
- Understand what is relevant and then make good use of that information to assess likely outcomes.
- Accelerate the pace of making informed decisions.

Learn more about data mining techniques in [Data Mining From A to Z](#), a paper that shows how organizations can use [predictive analytics](#) and data mining to reveal new insights from data.

Data Mining Techniques

1. Classification:

This analysis is used to retrieve important and relevant information about data, and metadata. This data mining method helps to classify data in different classes.

2. Clustering:

Clustering analysis is a data mining technique to identify data that are like each other. This process helps to understand the differences and similarities between the data.

3. Regression:

Regression analysis is the data mining method of identifying and analyzing the relationship between variables. It is used to identify the likelihood of a specific variable, given the presence of other variables.

4. Association Rules:

This data mining technique helps to find the association between two or more Items. It discovers a hidden pattern in the data set.

5. Outlier detection:

This type of data mining technique refers to observation of data items in the dataset which do not match an expected pattern or expected behavior. This technique can be used in a variety of domains, such as intrusion, detection, fraud or fault detection, etc. Outlier detection is also called Outlier Analysis or Outlier mining.

6. Sequential Patterns:

This data mining technique helps to discover or identify similar patterns or trends in transaction data for certain period.

7. Prediction:

Prediction has used a combination of the other data mining techniques like trends, sequential patterns, clustering, classification, etc. It analyzes past events or instances in a right sequence for predicting a future event.

Benefits of Data Mining:

- Data mining technique helps companies to get knowledge-based information.
- Data mining helps organizations to make the profitable adjustments in operation and production.
- The data mining is a cost-effective and efficient solution compared to other statistical data applications.
- Data mining helps with the decision-making process.
- Facilitates automated prediction of trends and behaviors as well as automated discovery of hidden patterns.
- It can be implemented in new systems as well as existing platforms
- It is the speedy process which makes it easy for the users to analyze huge amount of data in less time.

CHAPTER 4

4.1 Project Purpose and Scope

4.1 Purpose

The main aim of this project is to provide a reliable and secure cloud service and also increase trustworthiness of certifications by Continuous Automatic Auditing.

4.2 Project Scope

In MultiCloud environment, remote data integrity checking is required to secure user's data. User will upload file to Cloud. This file is split into blocks using Dynamic Block generation Algorithm. The Blocks are stored in Ternary Hash Tree (THT) format. The blocks have a parents node and child node. File Allocation Table (FAT) File System has proper Indexing and Metadata's for the different Chunks of the Cloud Storage. Here the auditor agrees to inspect logs, which are routinely created during monitoring operations by services providers to assess certification adherence. If Attacker corrupts data in MultiCloud, the continuous auditing process helps the verifier to perform Block and File level checking for remote data Integrity Checking using Verifiable Data Integrity Checking Algorithm. The auditing processes have a flow, first the parent block checking. If the parent block have any corrupted file then the child node auditing. If the child nodes have any corrupted file the File recovery is done by the Verifier automatically if the data gets corrupted during checking. Users can complaint cloud for file recovery.

4.3 Product Perspective

Cloud Computing enables the remote users to access data, services, and applications in on-demand from the shared pool of configurable computing resources, without the consideration of storage, hardware and software management. On the other hand, it is not easy for cloud users to identify whether Cloud Service Provider's (CSP) tag along with the data security legal expectations. So, cloud users could not rely on CSP's in terms of trust. So, it is significant to build a secure and efficient data auditing framework for increasing and maintaining cloud users trust with CSP. Researchers suggested introducing Third Party Auditor (TPA) on behalf of cloud user for verifying the outsourced data integrity, which may reduce the computation overhead of cloud users. In this work, we proposed a novel integrity verification framework for securing cloud storage based on Ternary Hash Tree (THT) and Replica based Ternary Hash Tree (R-THT), which will be used by TPA to perform data auditing. Differing from existing work, the proposed framework performs Block-level, File-level and Replica-level auditing with tree block ordering, storage block ordering for verifying the data integrity and ensuring data availability in the cloud. We further extend our framework to support error localization with data correctness, dynamic updates with block update, insert and delete operations in the cloud. The structure of THT and R-THT will reduce the computation cost and provide efficiency in data updates compared to the existing schemes. The security analysis of the proposed public auditing framework indicates the achievement of desired properties and performance has been evaluated

with the detailed experiment set. The results show that the proposed secure cloud auditing framework is highly secure and efficient in storage, communication and computation costs.

4.4 System Features

Public verification techniques enable users to outsource the data integrity verification to a dedicated third-party auditor. The auditor periodically checks the data integrity, and informs the users that the data may be corrupted once the checking fails [16]. In most of public verification schemes, the auditor is assumed to be honest and reliable. If the auditor is compromised, these schemes would be invalidated. For example, an irresponsible auditor may always generate a good integrity report without performing the verification to avoid the verification costs. In such a way, the auditor is virtually non-existent. Furthermore, a malicious auditor may collude with the cloud servers to generate a bias verification result to deceive the users for profits. To ensure the security in the case that the auditor is compromised, the users are required to audit the auditor's behaviors [17], [18], [19]: After each verification, the auditor records the information used to verify the data integrity, which enables the user to audit the validity of the auditor's behavior.

4.5 Design and Implementation Constraints

4.5.1 Constraints in Analysis

- ◆ Constraints as Informal Text
- ◆ Constraints as Operational Restrictions
- ◆ Constraints Integrated in Existing Model Concepts

- ◆ Constraints as a Separate Concept
- ◆ Constraints Implied by the Model Structure

4.5.2 Constraints in Design

- ◆ Determination of the Involved Classes
- ◆ Determination of the Involved Objects
- ◆ Determination of the Involved Actions
- ◆ Determination of the Require Clauses
- ◆ Global actions and Constraint Realization

4.5.3 Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

4.6 Other Nonfunctional Requirements

4.6.1 Performance Requirements

The application at this side controls and communicates with the following three main general components.

- ☐ embedded browser in charge of the navigation and accessing to the web service;
- ☐ Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

4.6.2 Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.
5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 5

5.1 Architecture Diagram:

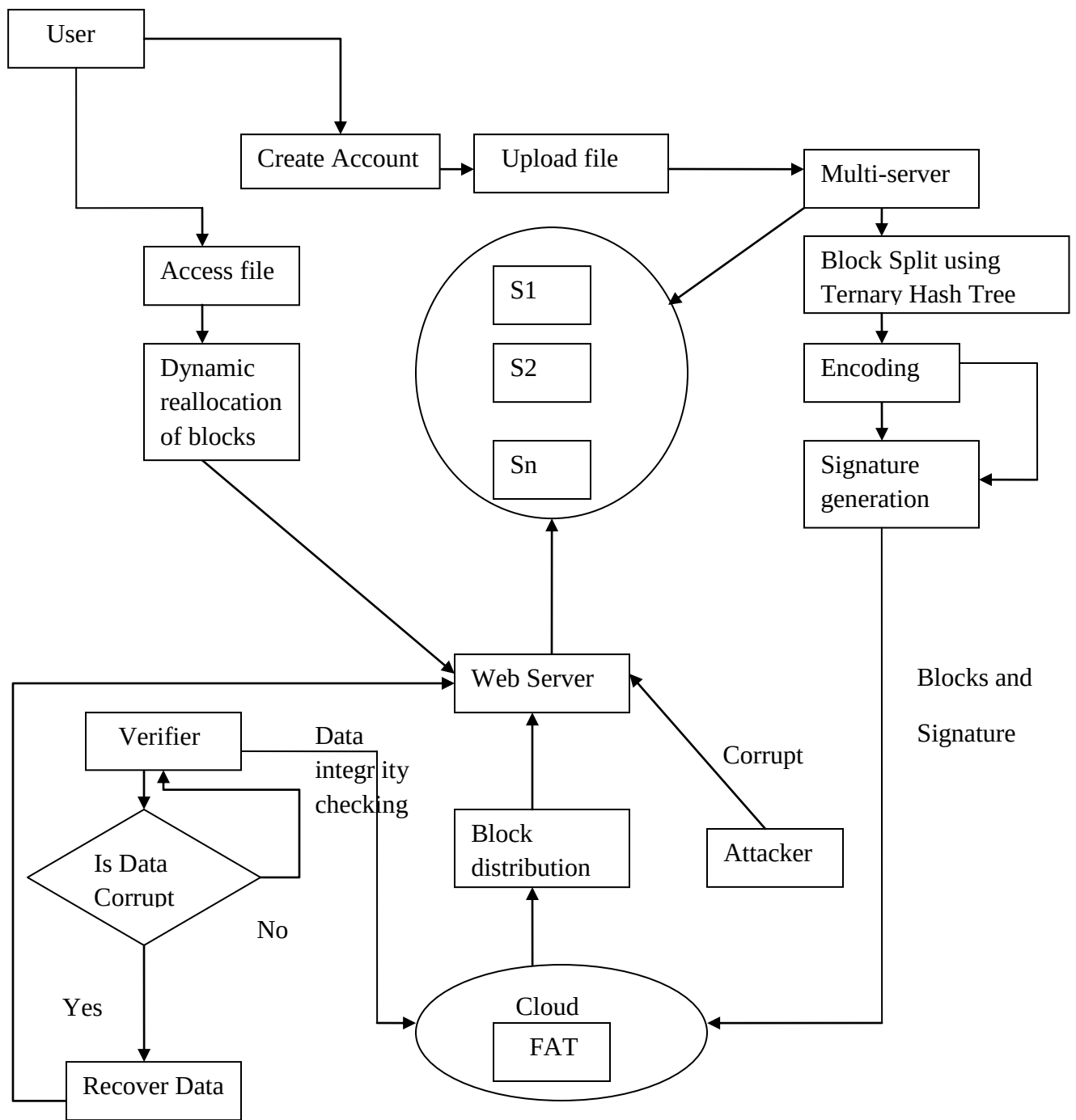
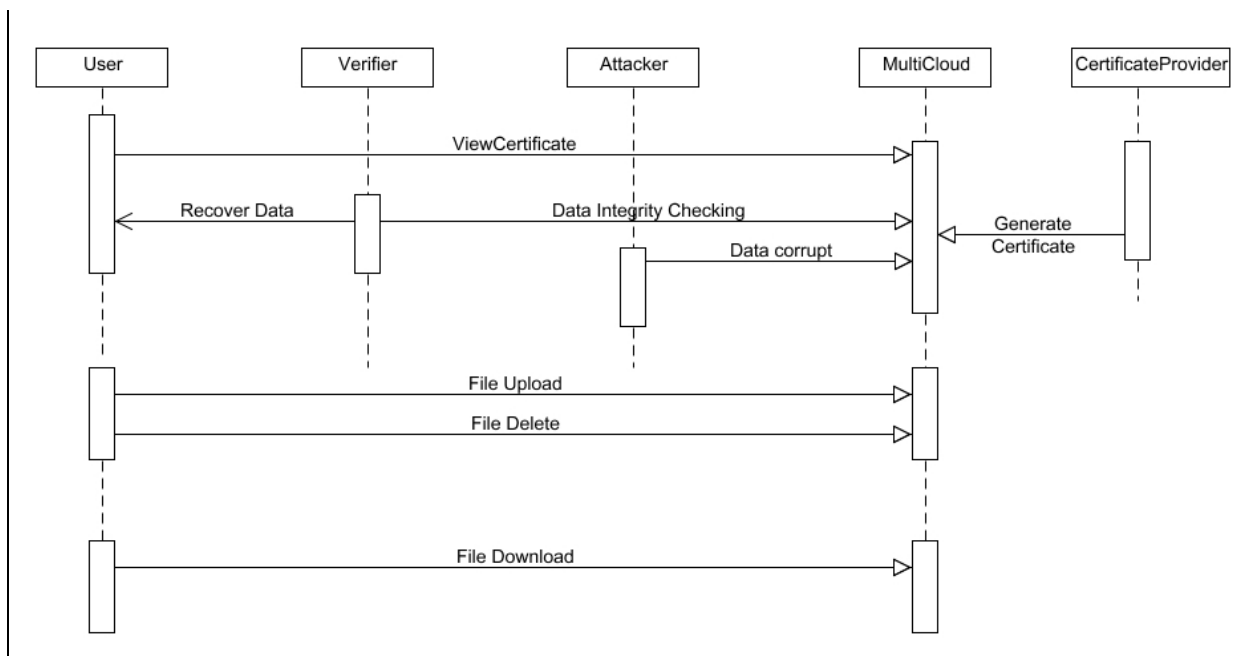


Fig: 5.1

5.2 Sequence Diagram:

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.



5.3 Use Case Diagram:

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

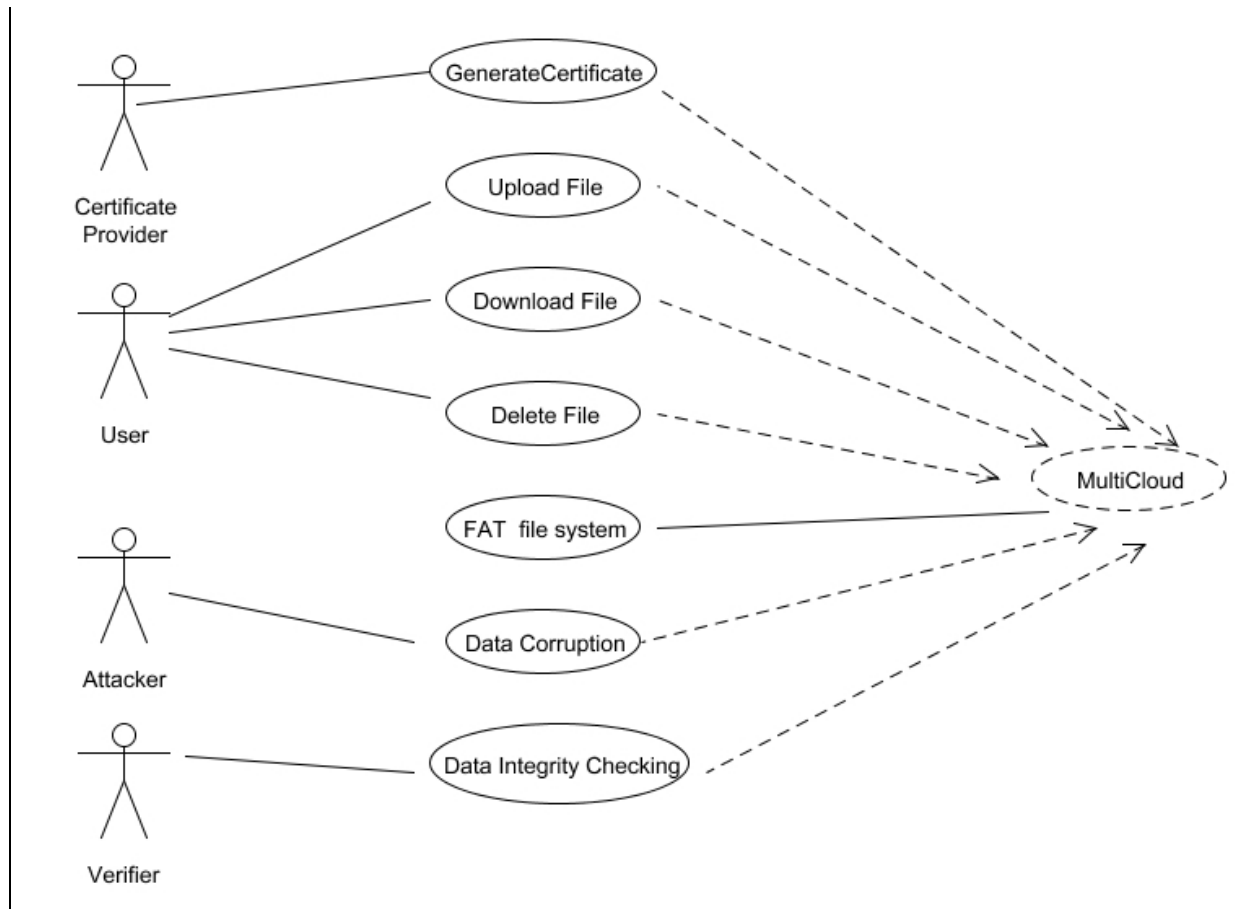
5.3.1. USECASE DIAGRAM

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

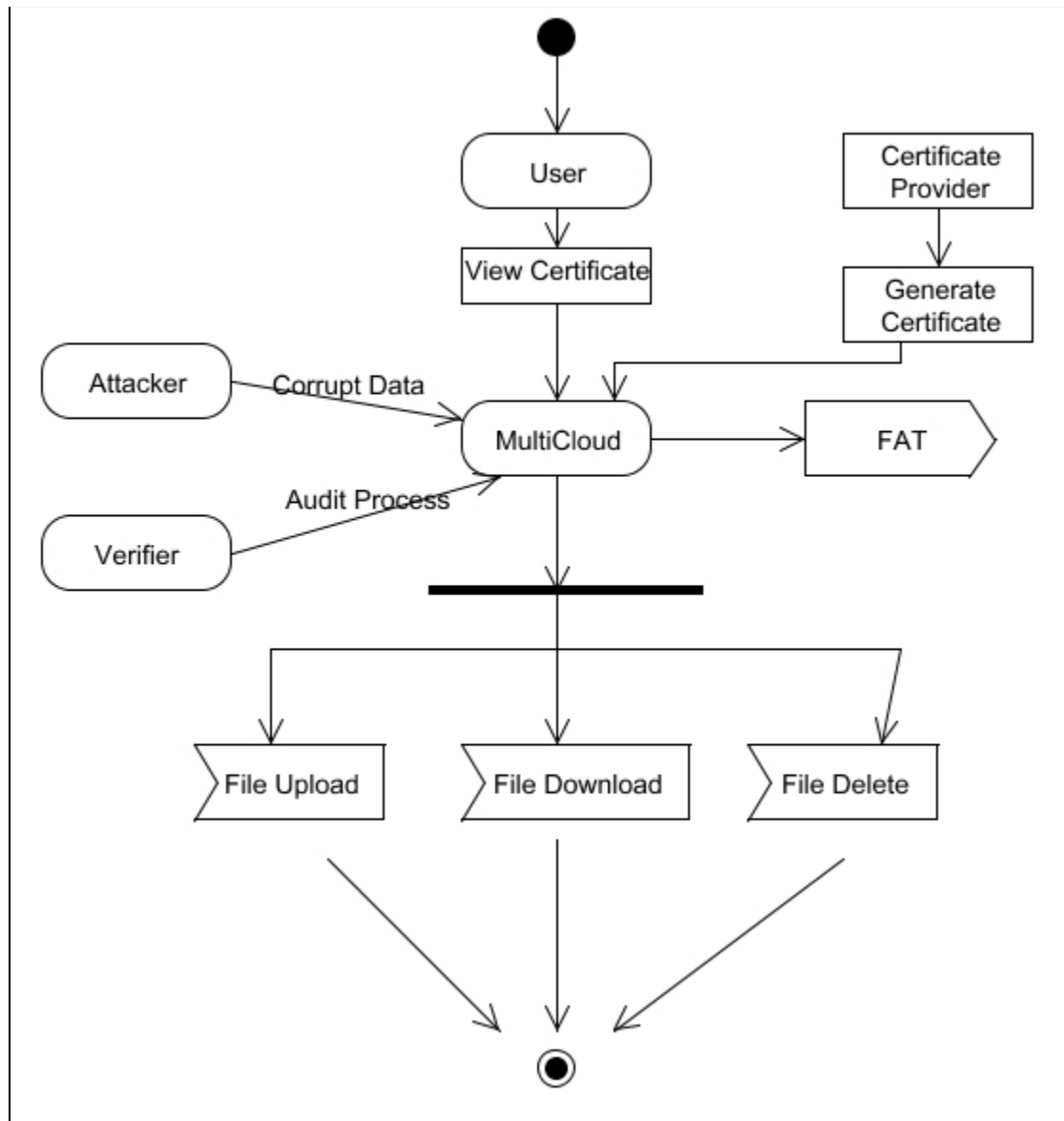


5.4 Activity Diagram:

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

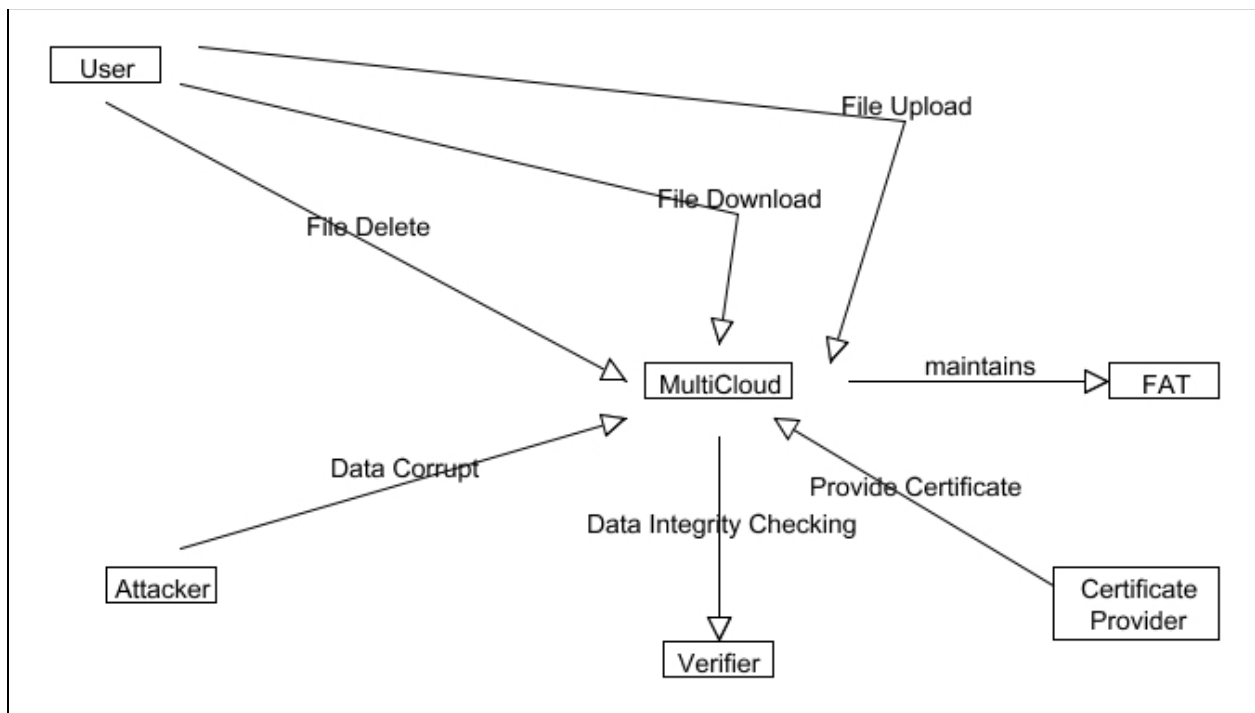
The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.



5.5 Collaboration Diagram:

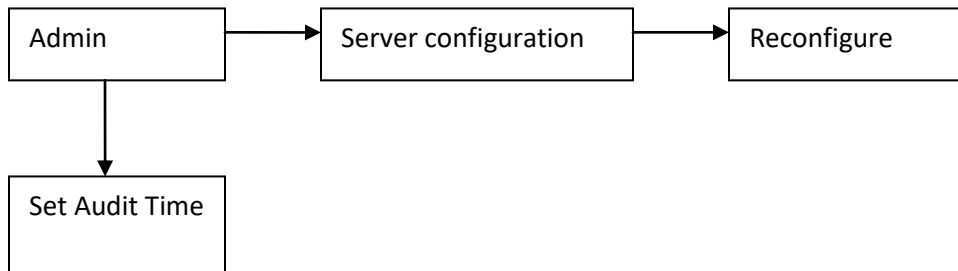
UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.



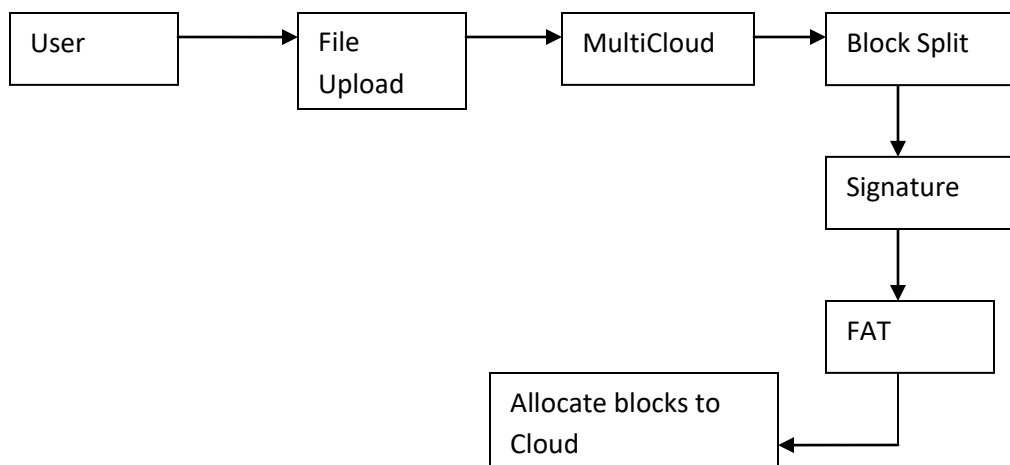
5.6 DATA FLOW DIAGRAM:

A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its aspects. It is a preliminary step used to create an overview of the system which can later be elaborated DFDs can also be used for visualization of data processing.

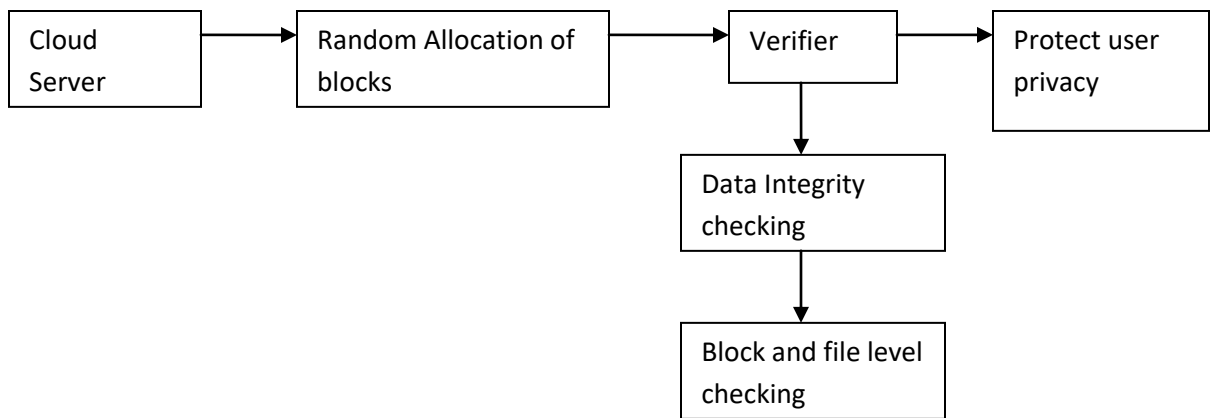
Level 0:



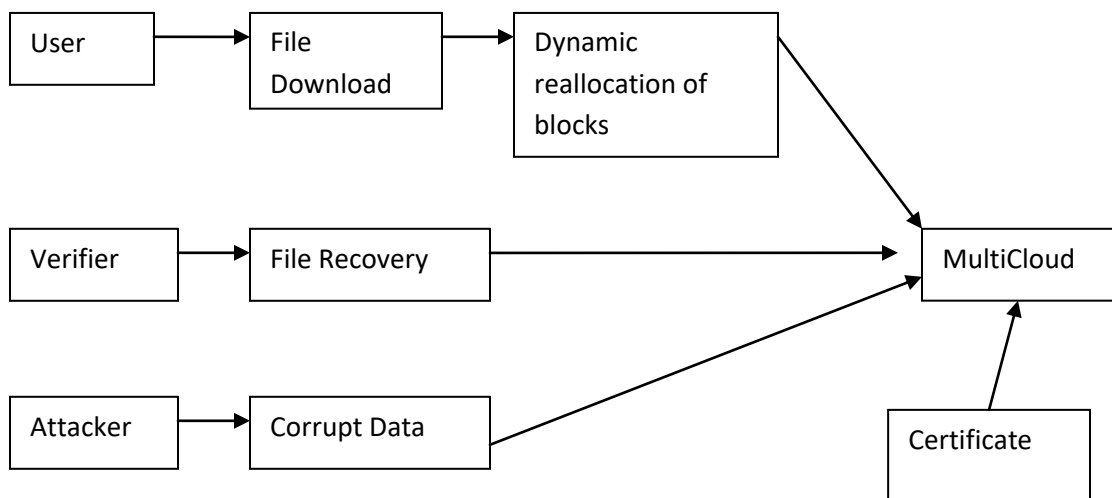
Level 1:



Level 2:

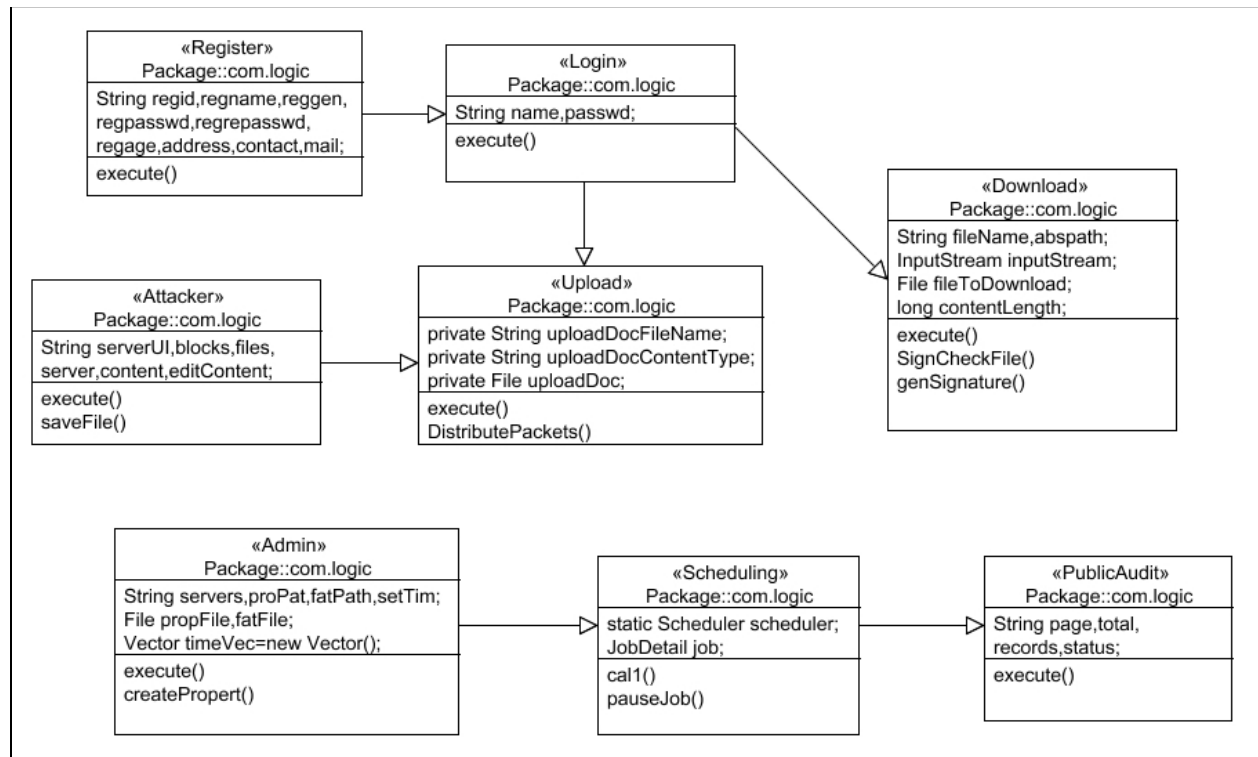


Level 3:



5.7 Class Diagram

A Class diagram in the Unified Modeling Language 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 objects.



CHAPTER 6

SYSTEM DESIGN

Admin configure Multi-Cloud server setup. Server IP Address and Port number is given by the admin for each Cloud. Now a Server Architecture is created for Multi-Cloud Storage. If the admin has to reconfigure the old Multi-Cloud server setup, it can be done. For old server setup, FAT file can be modified or remain same. Audit time will be set by the admin for Data Integrity checking process. User has an initial level Registration Process at the web end. The users provide their own personal information for this process. The server in turn stores the information in its database. After Registration, user can upload files to the server. Uploaded files will be stored in a Server. When the user upload the data to different cloud by the time it is splitted into different blocks using dynamic block generation Algorithm and each block will be appended with Signatures before storing the data in FATFS. Signature generated using MD5 Algorithm. Also the data gets encoded using for Base64 Algorithm. FATFS has proper Indexing and Metadata's for the different Chunks of the Data that is being uploaded by User. Verifier performs Remote Integrity Checking on Cloud Data. Cloud allocates random combination of all the blocks to the Verifier, instead of the whole file is retrieved during integrity checking. This is to protect user privacy from a third party (Verifier). Verifiable Data Integrity Checking Algorithm is done in two steps: Block Checking and File Checking. In Block Checking step: Three signatures are generated for Block level Checking.

- A signature of a block retrieved from a FATFS

- A new signature is generated for block to be checked
- A Signature is retrieved from the block appended with the signature which is stored in the Cloud

The above three signatures are cross checked for Block level Integrity Checking. And the block contents are appended to verify with File level Integrity Checking. Attacker can corrupt data in any one of the cloud servers. On Data Integrity Checking done by the Verifier, Verifier informs Corrupted blocks to the Cloud. Recovery Process will be done by the verifier automatically when data gets corrupted. User can complaint to the Cloud if the user file get corrupted (Verifier doesn't perform checking on this file). Whenever user access file, Blocks will be reallocated dynamically to provide access confidentiality in cloud and FAT File System will get updated. Auditor will monitor the cloud continuously and they provide the certificate based on the cloud performance. When new user join in the cloud they will read the certificate and then they can create an account in the cloud.

6.1 MODULES

- Server Configuration
- Data Upload and Block Split
- Data Integrity Checking and Update
- File Recovery

6.2 MODULE EXPLANATION:

6.2.1 Server Configuration

Admin configure Multi-Cloud server setup. Server IP Address and Port number is given by the admin for each Cloud. Now a Server Architecture is created for Multi-Cloud Storage. If

the admin has to reconfigure the old Multi-Cloud server setup, it can be done. For old server setup, FAT file can be modified or remain same. Audit time will be set by the admin for Data Integrity checking process.

Data Upload and Block Split

User has an initial level Registration Process at the web end. The users provide their own personal information for this process. The server in turn stores the information in its database. After Registration, user can upload files to the server. Uploaded files will be stored in a Server. When the user upload the data to different cloud by the time it is splitted into different blocks using dynamic block generation Algorithm and each block will be appended with Signatures before storing the data in FATFS. Signature generated using MD5 Algorithm. Also the data gets encoded using for Base64 Algorithm.

Data Integrity Checking and Update

FATFS has proper Indexing and Metadata's for the different Chunks of the Data that is being uploaded by User. Verifier performs Remote Integrity Checking on Cloud Data. Cloud allocates random combination of all the blocks to the Verifier, instead of the whole file is retrieved during integrity checking. This is to protect user privacy from a third party (Verifier). Verifiable Data Integrity Checking Algorithm is done in two steps: Block Checking and File Checking. In Block Checking step: Three signatures are generated for Block level Checking.

- ☐ A signature of a block retrieved from a FATFS
- ☐ A new signature is generated for block to be checked
- ☐ A Signature is retrieved from the block appended with the signature which is stored in the Cloud

The above three signatures are cross checked for Block level Integrity Checking. And the block contents are appended to verify with File level Integrity Checking.

File Recovery

Attacker can corrupt data in any one of the cloud servers. On Data Integrity Checking done by the Verifier, Verifier informs Corrupted blocks to the Cloud. Recovery Process will be done by the verifier automatically when data gets corrupted. User can complaint to the Cloud if the user file get corrupted (Verifier doesn't perform checking on this file).Whenever user access file, Blocks will be reallocated dynamically to provide access confidentiality in cloud and FAT File System will get updated. Auditor will monitor the cloud continuously and they provide the certificate based on the cloud performance. When new user join in the cloud they will read the certificate and then they can create an account in the cloud.

CHAPTER 7

CODING AND TESTING

7.1 CODING

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

7.2 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

7.2.1 NAMING CONVENTIONS

Naming conventions of classes, data member, member functions, procedures etc., should be **self-descriptive**. One should even get the meaning and scope of the variable by its name. The conventions are adopted for **easy understanding** of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

Class names

Class names are problem domain equivalence and begin with capital letter and have mixed cases.

Member Function and Data Member name

Member function and data member name begins with a lowercase letter with each subsequent letters of the new words in uppercase and the rest of letters in lowercase.

7.2.2 VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- ☐ Proper default values for the variables.
- ☐ Proper validation of values in the field.
- ☐ Proper documentation of flag values.

7.2.3 SCRIPT WRITING AND COMMENTING STANDARD

Script writing is an art in which indentation is utmost important. Conditional and looping statements are to be properly aligned to facilitate easy understanding. Comments are included to minimize the number of surprises that could occur when going through the code.

7.2.4 MESSAGE BOX FORMAT

When something has to be prompted to the user, he must be able to understand it properly. To achieve this, a specific format has been adopted in displaying messages to the user. They are as follows:

- ☐ X – User has performed illegal operation.
- ☐ ! – Information to the user.

7.3 TEST PROCEDURE

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing

during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- ☐ Static analysis is used to investigate the structural properties of the Source code.
- ☐ Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

7.4 TEST DATA AND OUTPUT

7.4.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

7.4.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- ☐ Performance Test

- ☐ Stress Test
- ☐ Structure Test

7.4.3 PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

7.4.4 STRESS TEST

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.4.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- ☐ Exercise all logical decisions on their true or false sides.
- ☐ Execute all loops at their boundaries and within their operational bounds.
- ☐ Exercise internal data structures to assure their validity.
- ☐ Checking attributes for their correctness.

- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

7.4.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.5 TESTING TECHNIQUES / TESTING STRATERGIES

7.5.1 TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful

test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

7.5.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- ☐ Flow graph notation
- ☐ Cyclometric complexity
- ☐ Deriving test cases
- ☐ Graph matrices Control

7.5.1.2 BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- ☐ Graph based testing methods
- ☐ Equivalence partitioning
- ☐ Boundary value analysis
- ☐ Comparison testing

7.5.2 SOFTWARE TESTING STRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- ☐ Testing begins at the module level and works “outward” toward the integration of the entire computer based system.

- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

7.5.2.1 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

7.5.2.2 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression. Condition testing method focuses

on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

7.5.2.3 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

7.5.2.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- * The function or performance characteristics confirm to specifications and are accepted.
- * A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving

deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

7.5.2.5 USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

- Input screen design.
- Output screen design.

Source Code

Screenshots:

REFERENCES

- [1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable Data Possession at Untrusted Stores,” Proc. 14th ACM Conf. Computer and Comm. Security (CCS’07), pp. 598-609, Oct. 2007.
- [2] M.A. Shah, M. Baker, J.C. Mogul, and R. Swaminathan, “Auditing to Keep Online Storage Services Honest,” Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HotOS ’07), pp. 1-6, 2007.
- [3] M.A. Shah, R. Swaminathan, and M. Baker, “Privacy-Preserving Audit and Extraction of Digital Contents,” Cryptology ePrint Archive, Report 2008/186, <http://eprint.iacr.org>, 2008.
- [4] Vladimir A. Dobrushkin, “Methods in Algorithmic Analysis”, CRC Press, 2009.

- [5] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, “Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing”, IEEE Transactions on Parallel and Distributed Systems, Vol. 22, No. 5, 847-859, 2011.
- [6] Hao, Z., Zhong, S., and Yu, N, “A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability”, IEEE Transactions on Knowledge and Data Engineering, Vol.23, 1432 – 1437, 2011.
- [7] Y. Zhu, G. J. Ahn, and Z. Hu, “Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds”, IEEE Transactions on Services Computing, Vol. 6, 227-238, 2011.
- [8] Wang, C., Wang, Q., Ren, K., Cao, N., and Lou, W, “Toward secure and dependable storage services in cloud computing”, IEEE Transactions on Services Computing, Vol.5, No.2, 220 – 232, 2012.
- [9] Cong Wang, Sherman S.M. Chow, Qian Wang, Kui Ren, and Wenjing Lou, “Privacy-Preserving Public Auditing for Secure Cloud Storage”, IEEE Transactions on Computers, Vol.62, No.2, 362-375, 2013.
- [10] Yang, K., and Jia, X, “An efficient and secure dynamic auditing protocol for data storage in cloud computing”, IEEE Transactions on Parallel and Distributed Systems, Vol.24, No.9, 1717 – 1726, 2013.
- [11] H. Wang, “Proxy Provable Data Possession in Public Clouds”, IEEE Transactions on Services Computing, Vol.6, No.4, 551 – 559, 2013.
- [12] Boyang Wang, Baochun Li, and Hui Li, “Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud”, IEEE Transactions on Cloud Computing, Vol.2, No.1, 2014.