Ensuring Data Storage Security in Cloud Computing

Project Report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Dr. Yashwant Singh

By

Akshay Gupta

111259

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "Ensuring Data Storage Security in Cloud Computing", submitted by Akshay Gupta in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date: 22 December, 2014 **Supervisor's Name** Dr. Yashwant Singh

Designation Assistant Professor (Senior Grade)

Acknowledgement

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of this project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

I express my warm thanks to my project guide Dr. Yashwant Singh from the Computer Science Department for their continuous support and guidance.

I would also like to thank the lab in-charge for their cooperation and all the people who provided me with the facilities being required and conductive conditions for my project

Date: 22-Dec-2014 Name of the student: Akshay Gupta

Table of Content

List of Figures

Abstract

Cloud computing is the best invention of the twenty first century. It has given a whole new structure to IT companies. It has changed the way employment works in the IT industry – it has generated a whole new set of careers such as from architects and developers to data scientists, security pros and more, all of which require a specific focus. And for the users, the increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centers.

Through this project we address the flip side of this important invention- Security. Even though cloud services offer plethora of benefits, still these benefits come with many security issues – like data corruption, loss etc. In this project we have used data encryption and duplicity techniques to offer data redundancy and correctness.

This system gives a reflective insight into the cloud security system. It is shown that how user maintains his resources, how new users are registered onto the system, how the admin uses the IP restriction process to filter the connections with the server, how the admin can manage and use his control panel for security purposes.

By using the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness

CHAPTER 1
INTRODUCTION

1.1 Introduction to Cloud Computing

Cloud computing is the delivery of computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage, social networking sites, webmail, and online business applications. The cloud computing model allows access to information and computer resources from anywhere that a network connection is available. Cloud computing provides a shared pool of resources, including data storage space, networks, computer processing power, and specialized corporate and user applications.[1]

The following definition of cloud computing has been developed by the U.S. National Institute of Standards and Technology (NIST): "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models."[1]

1.1.1 Characteristics

The characteristics of cloud computing include on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. On-demand self-service means that customers (usually organizations) can request and manage their own computing resources. Broad network access allows services to be offered over the Internet or private networks. Pooled resources means that customers draw from a pool of computing resources, usually in remote data centers. Services can be scaled larger or smaller; and use of a service is measured and customers are billed accordingly. [1]

1.1.2 Data Security

It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing. There are a number of security issues associated with cloud computing but these issues fall into two broad categories: security issues faced by cloud providers (organizations providing software-, platform-, or infrastructure via the cloud) and security issues faced by their customers (companies or organizations who host applications or store data on the on the cloud). The responsibility goes both ways, however: the provider must ensure that their infrastructure is secure and that their clients' data and applications are protected while the user must take measures to fortify their application and use strong passwords and authentication measures. [2]

1.1.3 Key Challenges

1.1.3.1 Data Breaches

The cloud security alliance has demonstrated that a virtual machine could use side-channel timing information to extract private cryptographic keys in use by other VMs on the same server. If a multitenant cloud service database isn't designed properly, a single flaw in one client's application could allow an attacker to get at not just that client's data, but every other clients' data as well. [3]

1.1.3.2 Data Loss

Data loss is the prospect of seeing your valuable data disappear into the ether without a trace. A malicious hacker might delete a target's data out of spite -- but then, you could lose your data to a careless cloud service provider or a disaster, such as a fire, flood, or earthquake. Compounding the challenge, encrypting your data to ward off theft can backfire if you lose your encryption key. [3]

1.1.3.3 Account Hijacking

If an attacker gains access to a client's credentials, they can eavesdrop on the client's activities and transactions, manipulate data, return falsified information, and redirect the client to illegitimate sites. The client's account or services instances may become a new

base for the attacker. From here, they may leverage the power of the cloud provider's reputation to launch subsequent attacks. [3]

1.1.4 Ensuring Security – What does it mean?

By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, our scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving servers. The first part of the section is devoted to a review of basic tools from coding theory that are needed in our scheme for file distribution across cloud servers. Then, the homomorphic token is introduced. The token computation function we are considering belongs to a family of universal hash function, chosen to preserve the homomorphic properties, which can be perfectly integrated with the verification of erasure-coded data. [4]

Subsequently, it is also shown how to derive a challenge response protocol for verifying the storage correctness as well as identifying misbehaving servers. Finally, the procedure for file retrieval and error recovery based on erasure-correcting code is outlined. [4]

1.2 PROBLEM STATEMENT

Why do we need security for the cloud?

Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood. Firstly, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging. Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion,

deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance. However, this dynamic feature also makes traditional integrity insurance techniques futile and entails new solutions. Last but not the least, the deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a robust and secure cloud data storage system in the real world. [4]

1.3 Objectives

To ensure the security and dependability for cloud data storage under the aforementioned adversary model, the objective is to design efficient mechanisms for dynamic data verification and operation and achieve the following goals –

1.3.3 Storage Correctness

To make sure that the data stored is correct and is monitored against any threats at all times. [4]

1.3.4 Fast Localization of Data Error

If any error is found in the user data, effectively locate the malfunctioning server when data corruption has been detected. [4]

1.3.5 Dynamic data support

To maintain the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud. [4]

1.3.6 Lightweight

To enable users to perform storage correctness checks with minimum overhead. [4]

1.4 Methodology

1.4.1 Theory

By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s). Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks, including: data update, delete and append. [4]

Data file F is dispersed among a set of N=m+k distributed servers. A (m + k, k) Reed-Solomon erasure-correcting code is used to create k redundancy parity vectors from m data vectors in such a way that the original m data vectors can be reconstructed from any m out of the m + k data and parity vectors. By placing each of the m + k vectors on a different server, the original data file can survive the failure of any k of the m+k servers without any data loss, with a space overhead of k/m. For support of efficient sequential I/O to the original file, our file layout is systematic, i.e., the unmodified m data file vectors together with k parity vectors is distributed across m+ k different servers. [4]

In order to achieve assurance of data storage correctness and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. The main idea is as follows: before file distribution the user pre-computes a certain number of short verification tokens on individual vector G(j) ($j \in \{1, \ldots, n\}$), each token covering a random subset of data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens pre-computed by the user. [4]

1.5 Report Order/Organization

Chapter – 2 Literature Survey

Here, the subject 'Security in Cloud Computing' is researched thoroughly and all the theory that can help the user understand the basis of the system is given. Cloud

computing, its architecture, deployment models, service models, database connectivity – the topics which constitute the basic pillars of the project are explained.

Chapter – 3 Analysis and Design

Here, first the security system is introduced with its basic principle of homomorphic token. Then the Software Requirement Specification is presented to lay out the functional and non-functional requirements of the system. Here various design constraints, user classes, system features and interfaces are talked about.

Chapter – 4 Implementation

In this part, the basic pillars of the system, like the database connectivity, cloud simulation etc. are presented in their practical usage form i.e. the codes are given to make the reader understand about the practical product taken out of the literature survey.

Chapter – 5 Conclusion

Finally, a summary of the whole report is presented to highlight all the key points of the work in brief. Here we depict what we have done and what is to be done. Cloud computing security is still in its initial stage and hence, further work is emphasized in the conclusion. However, this project serves as a solid model for any research on the security systems for cloud computing.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

Cloud computing is a computing paradigm, where a large pool of systems are connected in private or public networks, to provide dynamically scalable infrastructure for application, data and file storage. With the advent of this technology, the cost of computation, application hosting, content storage and delivery is reduced significantly. Cloud computing is a practical approach to experience direct cost benefits and it has the potential to transform a data center from a capital-intensive set up to a variable priced environment. The idea of cloud computing is based on a very fundamental principle of 'reusability of IT capabilities'. The difference that cloud computing brings compared to traditional concepts of "grid computing", "distributed computing", "utility computing", or "autonomic computing" is to broaden horizons across organizational boundaries.

Cloud Computing has been envisioned as the next generation architecture of IT Enterprise. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers. [4][5]

2.2 The Java Programming Language

2.2.1 Introduction

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. [6]

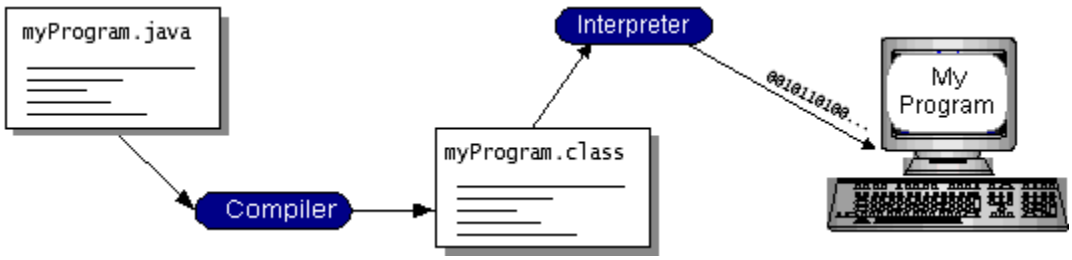

Fig 1 Compiler & Interpreter

Ref - http://www.javaguicodexample.com/javainstalltestusejdk.html

You 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. [6]

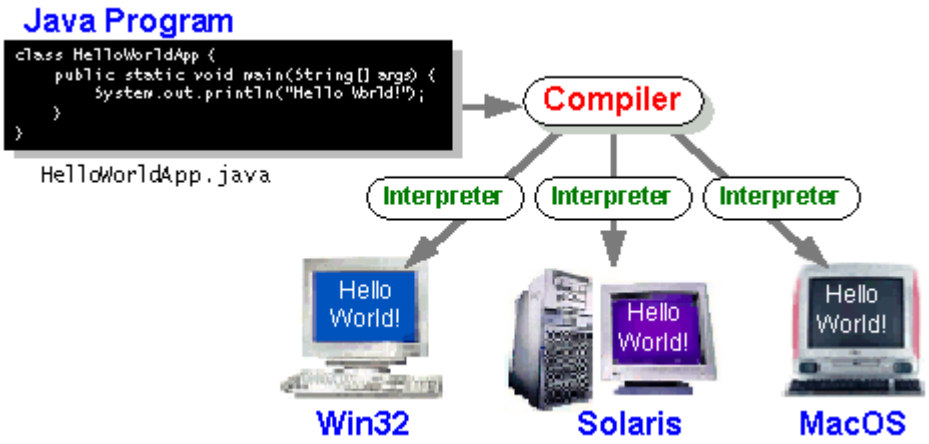

Fig 2 Compiling a program

Ref- http://www.math.uni-hamburg.de/doc/java/tutorial/getStarted/cupojava/unix.html

2.2.2 The Java Platform

A *platform* is the hardware or software environment in which a program runs. Some of the most popular platforms are 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. [6]

The Java platform has two components:

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

Java VM is the base for the Java platform and is ported onto various hardware-based platforms. [6]

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. [6]

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. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability. [6]

2.2.3 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. [6]

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 popular 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. [6]

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features: [6]

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on. [6]
- **Applets**: The set of conventions used by applets. [6]
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses. [6]
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language. [6]
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates. [6]
- **Software components**: Known as JavaBeans™, can plug into existing component architectures. [6]
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI). [6]
- **Java Database Connectivity (JDBC™)**: Provides uniform access to a wide range of relational databases. [6]

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK. [6]

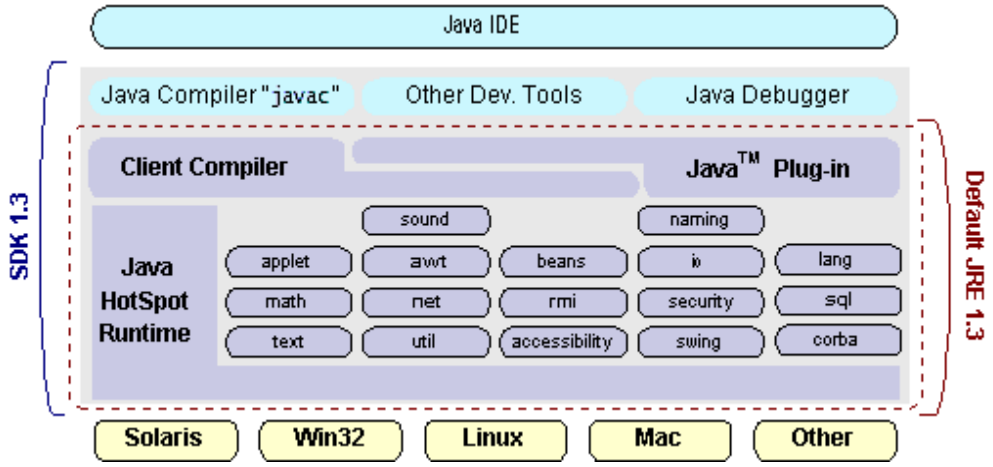

Fig 3 Java 2 SDK Constituents

Ref - http://www.math.uni-hamburg.de/doc/java/tutorial/getStarted/intro/cando.html

2.3 Connectivity

2.3.1 ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change. [7]

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures

might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN. [7]

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources. [7]

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer. [7]

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year. [7]

2.3.2 JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on. [7]

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution. [7]

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

2.3.3 JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java. [7]

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. ***SQL Level API***

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application

programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user. [7]

2. ***SQL Conformance***

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users. [7]

3. ***JDBC must be implemental on top of common database interfaces***

The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa. [7]

4. ***Provide a Java interface that is consistent with the rest of the Java system***

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system. [7]

5. ***Keep it simple***

6. This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API. [7]

7. ***Use strong, static typing wherever possible***

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime. [7]

8. ***Keep the common cases simple***

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible. Finally we decided to proceed the implementation using Java Networking. And for dynamically updating the cache table we go for MS Access database. **Java has two things: a programming language and a platform.** [7]

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer. [7]

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

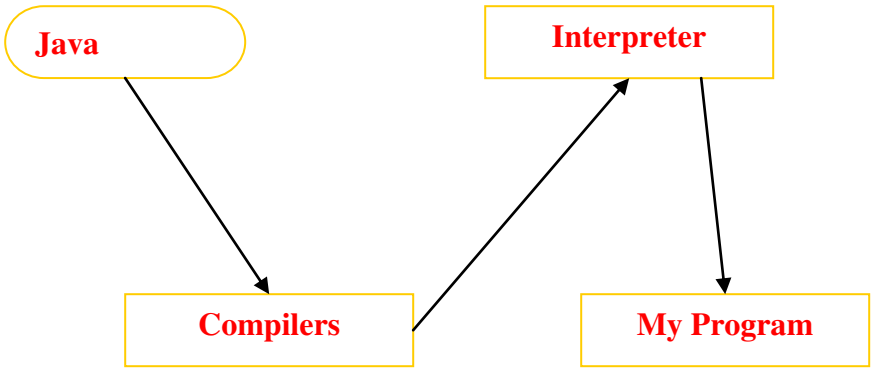

Fig 4 - Compiler and interpreter

Ref - http://www.javaguicodexample.com/javainstalltestusejdk.html

We 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 Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware. [7]

Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh. [7]

2.4 Architecture

Cloud Computing Service Models Cloud Providers offer services that can be grouped into three categories.

2.4.1 Service Models

1. **Software as a Service (SaaS)**: In this model, a complete application is offered to the customer, as a service on demand. A single instance of the service runs on the cloud & multiple end users are serviced. On the customers' side, there is no need for upfront investment in servers or software licenses, while for the provider, the costs are lowered, since only a single application needs to be hosted & maintained. Today SaaS is offered by companies such as Google, Salesforce, Microsoft, Zoho, etc. [8]

2. **Platform as a Service (Paas)**: Here, a layer of software, or development environment is encapsulated & offered as a service, upon which other higher levels of service can be built. The customer has the freedom to build his own applications, which run on the provider‟s infrastructure. To meet manageability and scalability requirements of the applications, PaaS providers offer a predefined combination of OS and application servers, such as LAMP platform (Linux, Apache, MySql and PHP), restricted J2EE, Ruby etc. Google‟s App Engine, Force.com, etc are some of the popular PaaS examples. [8]

3. **Infrastructure as a Service (Iaas)**: IaaS provides basic storage and computing capabilities as standardized services over the network. Servers, storage systems, networking equipment, data centre space etc. are pooled and made available to handle workloads. The customer would typically deploy his own software on the infrastructure. Some common examples are Amazon, GoGrid, 3 Tera, etc. [8]

2.4.2 Deployment Models

Enterprises can choose to deploy applications on Public, Private or Hybrid clouds. Cloud Integrators can play a vital part in determining the right cloud path for each organization.

Public Cloud - Public clouds are owned and operated by third parties; they deliver superior economies of scale to customers, as the infrastructure costs are spread among a mix of users, giving each individual client an attractive low-cost, "Pay-as-you-go" model. All customers share the same infrastructure pool with limited configuration, security protections, and availability variances. These are managed and supported by the cloud provider. One of the advantages of a Public cloud is that they may be larger than an enterprises cloud, thus providing the ability to scale seamlessly, on demand. [8]

Private Cloud - Private clouds are built exclusively for a single enterprise. They aim to address concerns on data security and offer greater control, which is typically lacking in a public cloud. There are two variations to a private cloud: -
On-premise Private Cloud: On-premise private clouds, also known as internal clouds are hosted within one's own data center. This model provides a more standardized process and protection, but is limited in aspects of size and scalability. IT departments would also need to incur the capital and operational costs for the physical resources. This is best suited for applications which require complete control and configurability of the infrastructure and security. – Externally hosted Private Cloud: This type of private cloud is hosted externally with a cloud provider, where the provider facilitates an exclusive cloud environment with full guarantee of privacy. This is best suited for enterprises that don't prefer a public cloud due to sharing of physical resources. [8]

Hybrid Cloud - Hybrid Clouds combine both public and private cloud models. With a Hybrid Cloud, service providers can utilize 3rd party Cloud Providers in a full or partial manner thus increasing the flexibility of computing. The Hybrid cloud environment is capable of providing on-demand, externally provisioned scale. The ability to augment a private cloud with the resources of a public cloud can be used to manage any unexpected surges in workload. [8]

2.5 Security Issues

Cloud computing is an emerging technology with shared resources, lower cost and rely on pay per use according to the user demand. Due to many characteristics it has effect on

IT budget and also impact on security, privacy and security issues .In this section all these issues are discussed. All those CSPs who wish to enjoy this new trend should take care of these problems. As Pakistan is developing country with no any proper IT strategy, a CSP should give their full attention to security aspect of cloud because it is a shared pool of resources. Customer not know where the data are stored, who manage data and other vulnerabilities that can occur. [9]

Following are some issues that can be faced by CSP while implementing cloud services.

2.5.1 Privacy Issue - It is the human right to secure his private and sensitive information. In cloud context privacy occur according to the cloud deployment model. In Public cloud (accessed through the Internet and shared amongst different consumers) is one of the dominant architecture when cost reduction is concerned, but relying on a CSP to manage and hold customer information raises many privacy concerns and are discussed under. [9]

2.5.2 Lack of user control - In SAAS environment service provider is responsible to control data. Now how customer can retain its control on data when information is processed or stored. It is legal requirement of him and also to make trust between customer and vendor. In this new paradigm user sensitive information and data is processed in 'the cloud' on systems having no any, therefore they have danger of misuse, theft or illegal resale. [9]

4.1.2 Unauthorized Secondary - Usage One of the threats can occur if information is placed for illegal uses. Cloud computing standard business model tells that the service provider can achieve profits from authorized secondary uses of users' data, mostly the targeting of commercials. Now a days there are no technological barriers for secondary uses. In addition, it has the connected issue of financial flexibility of the CSPs: for example, possibility of vendor termination, and if cloud computing provider is bankrupted or another company get data then what would happen. [9]

4.1.3 Transborder Data Flow and Data Proliferation - One of the attribute of cloud is Data proliferation and which involves several companies and is not controlled and managed by the data owners. Vendor guarantee to the ease of use by copy data in several datacenters. This is very difficult to ensure that duplicate of the data or its backups are not stored or processed in a certain authority, all these copies of data are deleted if such a request is made. Due to movement of data, CP exacerbate the transborder data flow

matter because it can be tremendously difficult to ascertain which specific server or storage device will be used, as the dynamic nature of this technology. [9]

4.1.4 **Dynamic provision -** Cloud has vibrant nature so there is no clear aspect that which one is legally responsible to ensure privacy of sensitive data put by customer on cloud. [9]

CHAPTER – 3
SYSTEM ANALYSIS AND DESIGN

3.1 INTRODUCTION

By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server. Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks, including: data update, delete and append.[4]

3.2 SOFTWARE REQUIREMENT SPECIFICATION

1.1 Purpose

This document is a Software Requirement Specification for the project 'Ensuring Data Storage Security in Cloud Computing'. The software is designed so as to meet the various design goals such as storage correctness, error localization etc.

1.2 Intended Audience and Reading Suggestions

This project is produced as a initial step for developing a full-fledged security software. It covers all the functionalities of the security system based on a simulated cloud environment. This is for anyone who wants to read some quality research in this field.

1.3 Product Scope

This project is strictly for testing purposes and need to be done on a large scale for its use in commercial purposes. However, it can provide as a good benchmark for developing a commercial security system.

This project has used cloud simulation to establish a data security system. For commercial production of this software, a real life cloud platform will be needed where large data centers hold the information of users within the cloud environment.

2. Overall Description

2.1 Product Perspective

Cloud Computing has been envisioned as the next generation architecture of IT Enterprise. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood. In this project, we focus on cloud data storage security, which has always been an important aspect of quality of service. [4]

2.2 Product Functions

1. User account maintenance – Login, Add, delete or modify data functions.
2. New user registration.
3. Hacker's information records.
4. IP filtering. (Keeping restricted IP address list)
5. Mobile Alert Service

2.3 User Classes and Characteristics

Users of this cloud based integrated development environment will mainly be software developers and research professionals. Since it is reasonable to assume that an average developer has knowledge about functionalities and usage of Eclipse or other java platforms, we assume that our users will already be informed about basic functionality of

the product. Also clear documentation and tutorials about the product feature will be provided.

2.4 Operating Environment

Hardware Requirements:

- System Pentium IV 2.4 GHz.
- Hard Disk 40 GB.
- Floppy Drive 1.44 Mb.
- Monitor 15 VGA Colour.
- Mouse Logitech.
- Ram 256 Mb.

Software Requirements:

i. Operating system Windows XP Professional
ii. Front End JAVA, Swing(JFC),RMIJ2M
iii. Back End MS-Access
iv. Tool Eclipse 3.3

2.5 Design and Implementation Constraints

Developers of the product should be aware that main feature of the intended product is security. So they should use common libraries and tools that can work with all the java platform with no problem.

Developers should be careful about the privacy of users. Since product will be cloud application, all user data will be kept on cloud server and necessary precautions should be taken to protect user data.

Since product will be cloud application and all user programs will be executed on cloud server, developers should limit the privileges of the users so that they cannot harm other users' data and system server.

2.6 User Documentation

Along with the software product, this report is sufficient enough to help people understand the working methodology and usage of the developed prototype system.

2.7 Assumptions and Dependencies

The product would build on leveraging existing systems of cloud computing. In this regard, necessary inspirations could be obtained by analyzing related systems such as Amazon Web Services(AWS),

The user must understand the real cloud environment is entirely different and here only a simulation of the cloud is shown. Although the basic security structure will be the same in both cases. And the security system will be applicable on a real cloud too.

The user must have basic understanding of a Java Integrated Development Tool like Eclipse.

3. External User Interfaces

3.1 User Interfaces

This group of requirements is related to external interaction of the workspace with outer world. For user to interact with the workspace, product will provide both command line interface and graphical interface. Command line interface will be UNIX like and graphical interface will allow tabbed navigation of windows, hierarchical view of workspace etc.

Through the GUI the user would be able to create a new account or review data. And command prompt would enable the admin to work with the system configuration.

3.2 Software Interfaces

3.2.1 Command Line Support

Except the graphical user interface, system will provide a command line support for the users. Although most of the functionality will be served on graphical user interface, some complex tasks may be accomplished through the command line as in the Linux OS distributions. [10]

To use the command line interface, user first should login to the system with his own credentials. After login user will be redirected to the main page of the Cloud IDE. In this main page user can go to the below part of the page which will be a command line where user can execute UNIX commands on the current directory. [10]

3.2.2 Project/Workspace Explorer

Project/Workspace explorer is going to be the main user interface of the web based integrated development environment. User will be able to their file hierarchy and edit it.
To be able to use this main graphical user interface, user should have login to the system. After login, user will be directed to a main page specialized for himself. In this main interface there will be a workspace explorer where users can see his file hierarchy and create and edit the files in workspace. [10]

4. System Features

In this subsection, we will examine the features of the system in detail by categorizing them according to their functionality. [10]

4.1 User Authentication

4.1.1 Description

Product will be used via eclipse IDE tool. Each user must be logged in to the server to access his workspace. Hence, first-time users must be complete registration process. To register to the system, user must specify some information asked during registration.

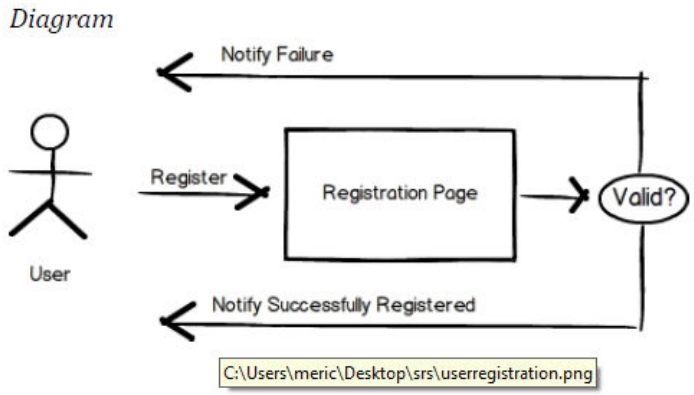

Figure 5 – User Registration

Self-made

After validation, registration will completed, and user will be informed.

4.1.2 Normal Flow of Events

1. User opens the registration page
2. User specifies his information
3. System validate the specified information
4. User is registered to the system

4.1.3 Alternative Event Flow 1

1. User can not registered to the system due to inappropriate information

4.2 User login

4.2.1 Explanation via diagram

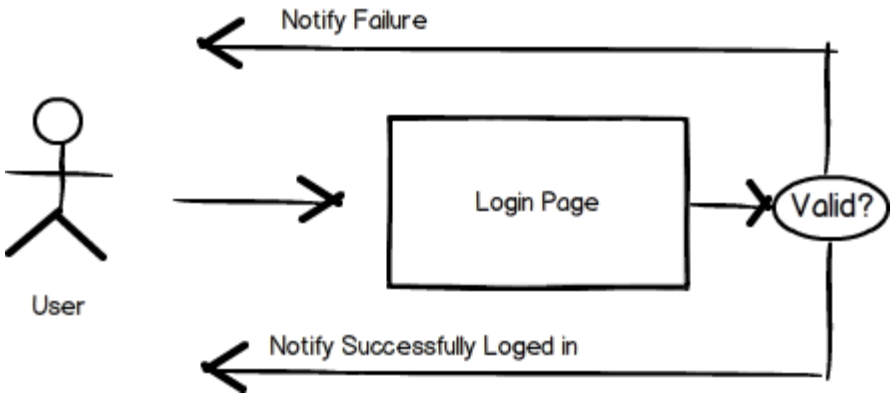

Figure 6 – User Login Diagram

Self-made

4.2.2 Normal Flow of Events

1. User opens the login page
2. User tries to login to the system with his credentials
3. System validate the specified information
4. User is logged into the system.

4.2.3 Alternative Event Flow

User cannot logged into the system due to incorrect credentials

4.3 Create File/Folder

4.3.1 Explanation via diagram

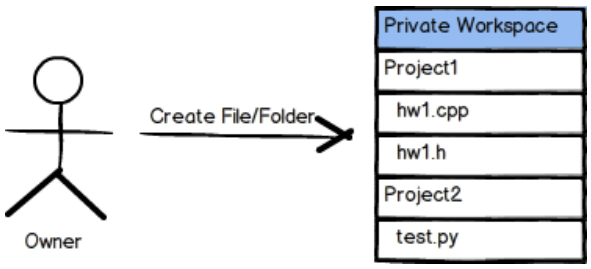

Figure 7 – Create File/Folder

Self-made

4.3.2 Normal Flow of Events

1. User logins in to the system
2. User opens his workspace
3. User creates a folder
4. User opens newly created folder
5. User creates a file

4.4 IP Filtering

4.4.1 Explanation via Diagram

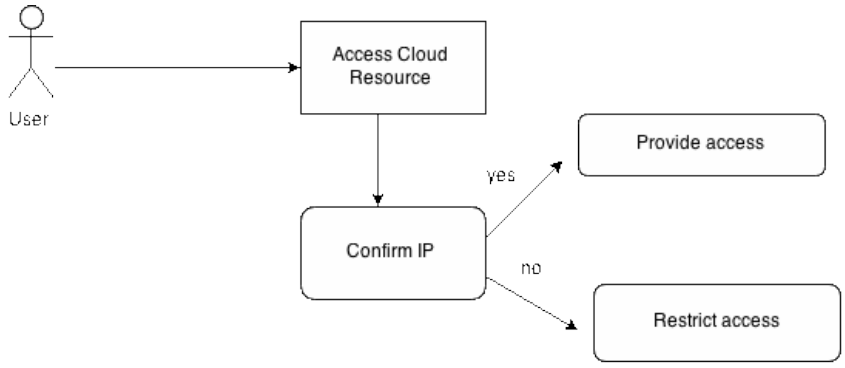

Figure – 8 IP Filtering

Self-made

4.4.2 Normal flow of events

- User tries to access the system.
- Server checks the user IP and filters it using restricted IP records.
- If the IP is safe, it is granted access.
- If it is malicious, a warning is generated.

4.5 Mobile Alert Service

4.5.1 Description and diagram

Anytime a user makes a service request, there will be an alert generated for the administrator mobile device. This functionality is implemented via simulator in the security system.

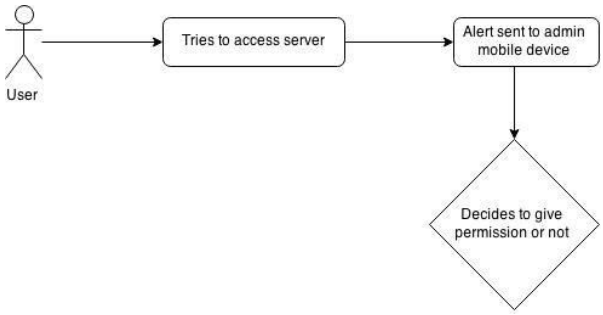

Figure – 9 Mobile alert Service

Self-made

4.5.2 Normal Flow of Events

- User tries to access the security system.
- His details are sent to the mobile device.
- Admin choses to allow/block the user.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Server machine should have a powerful CPU and high speed internet access so that it can handle multiple users at the same time. Another performance requirement is the storage space. Higher storage space means more user and bigger workspace per user so higher the storage, better the performance. [10]

Performance requirement by the user side is, web application should be developed as a lightweight web app so that it can work on almost any platform even with slower internet connections. [10]

5.2 Security Requirements

Since this software will be hosted on cloud server, all the user data will be kept on the cloud server. Product should be able to protect privacy of user data. Workspace of the user should only be accessed through user own credentials and any other user should not be able to access to the user private data. Since execution will also be done in the machine in the cloud, user should be restricted in terms of user rights. User should only access to his own workspace and should not access to any other workspace with the programs they run on the cloud. Also rights of the user should be restricted so that user can not harm to system by the programs they run or by the commands they run on terminal. [10]

Since all the data will be transferred on the cloud server, system should also use an encryption and decryption mechanism only intended user can decode the data and work on the data. [10]

3.3 Architectural Diagram

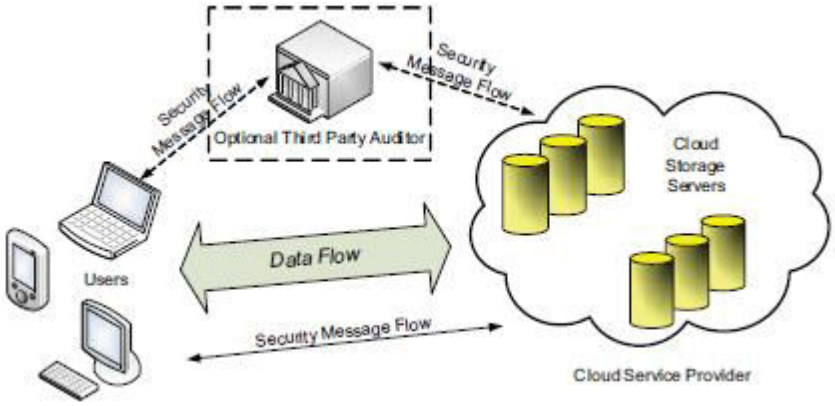

Fig 10 – Architectural Diagram

Ref - https://**eprint**.iacr.org/2009/081.pdf

Description –

In the above diagram it is shown how the user interacts with the cloud platform. First the user logs onto the cloud server with his credentials. The server reverts back and provides access to the user. After that the data flow occurs between the user and the cloud server. In case of any verification procedures or discrepancies found, the third party auditor can check the user and server security configurations.

3.4 Data Flow Diagram

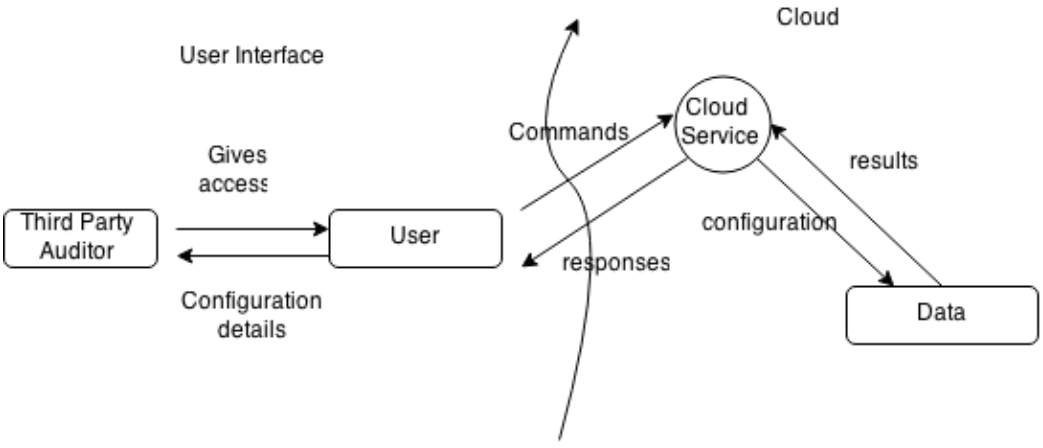

Fig-11 Data Flow Diagram

Description –

The above diagram aptly displays that the user interacts with the cloud server to retrieve its data or perform any modifications. The cloud service sends the user access controls like add, delete or modify the data. For that it sends the user configuration to the database so that necessary data can be fetched. The TPA can check user records for security purposes.

3.5 UML DIAGRAM

3.5.1 Control Flow Diagram

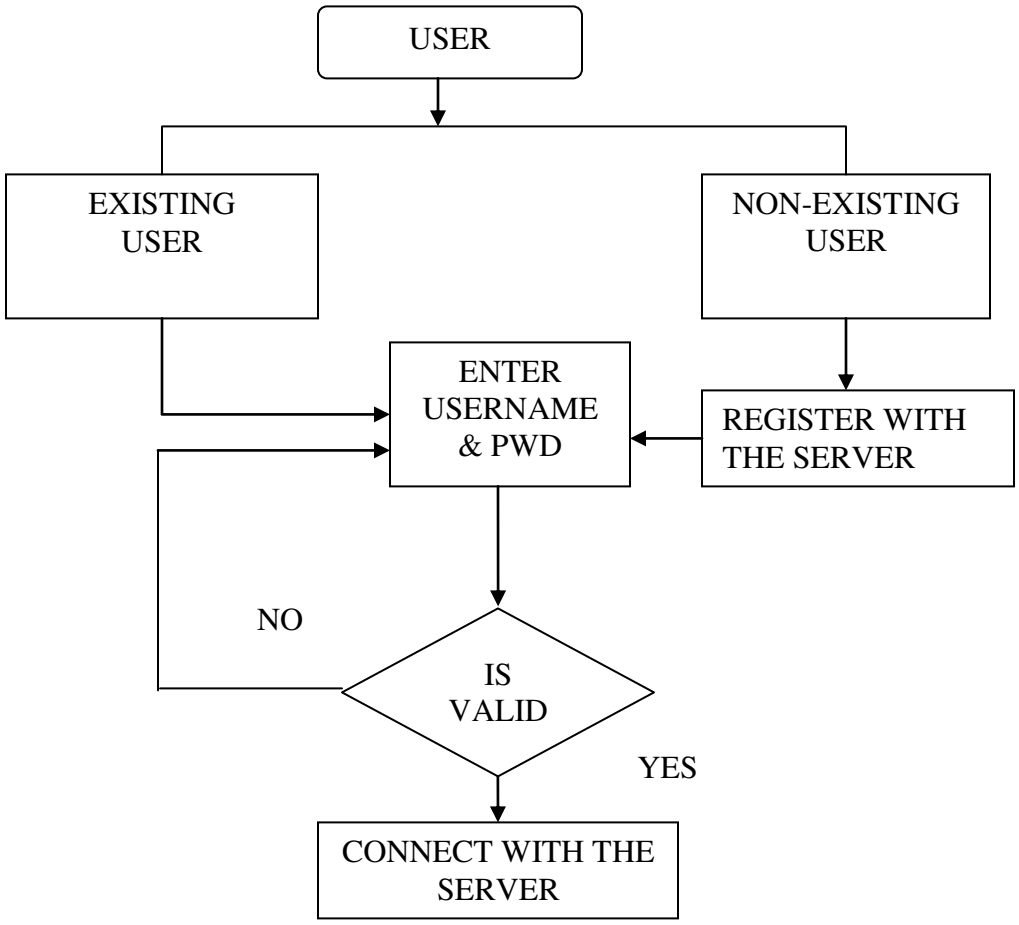

Fig 12 – **Control Flow Diagram**

Ref – Self-made

Description – As shown in the diagram the user interacts with login panel and new user is allowed to register with the system. Once the login is done, the user is free to add, delete or modify data. And obviously, this is accomplished by connecting with the server.

3.5.2 ACTIVITY DIAGRAM

An activity diagram is characterized by states that denote various operations. Transition from one state to the other is triggered by completion of the operation. The purpose of an

activity is symbolized by round box, comprising the name of the operation. An operation symbol indicates the execution of that operation. This activity diagram depicts the internal state of an object.

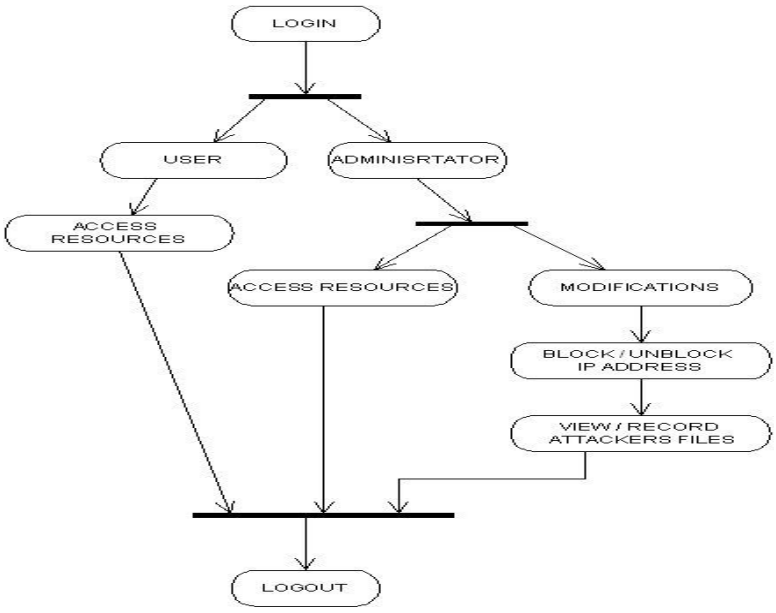

Fig.13 Activity Diagram

Ref – Self made

Description

As we can see from the diagram, two login situations can occur – either the user will login or the admin will. In case of user, the flow of actions are login and then access to its resources.

While in case of the admin, the actions are resources access as well as modifications in the database. In case the admin wants to modify the database or configuration, he can use the 'Block/Unblock IP option)

3.5.3 Sequence Diagram

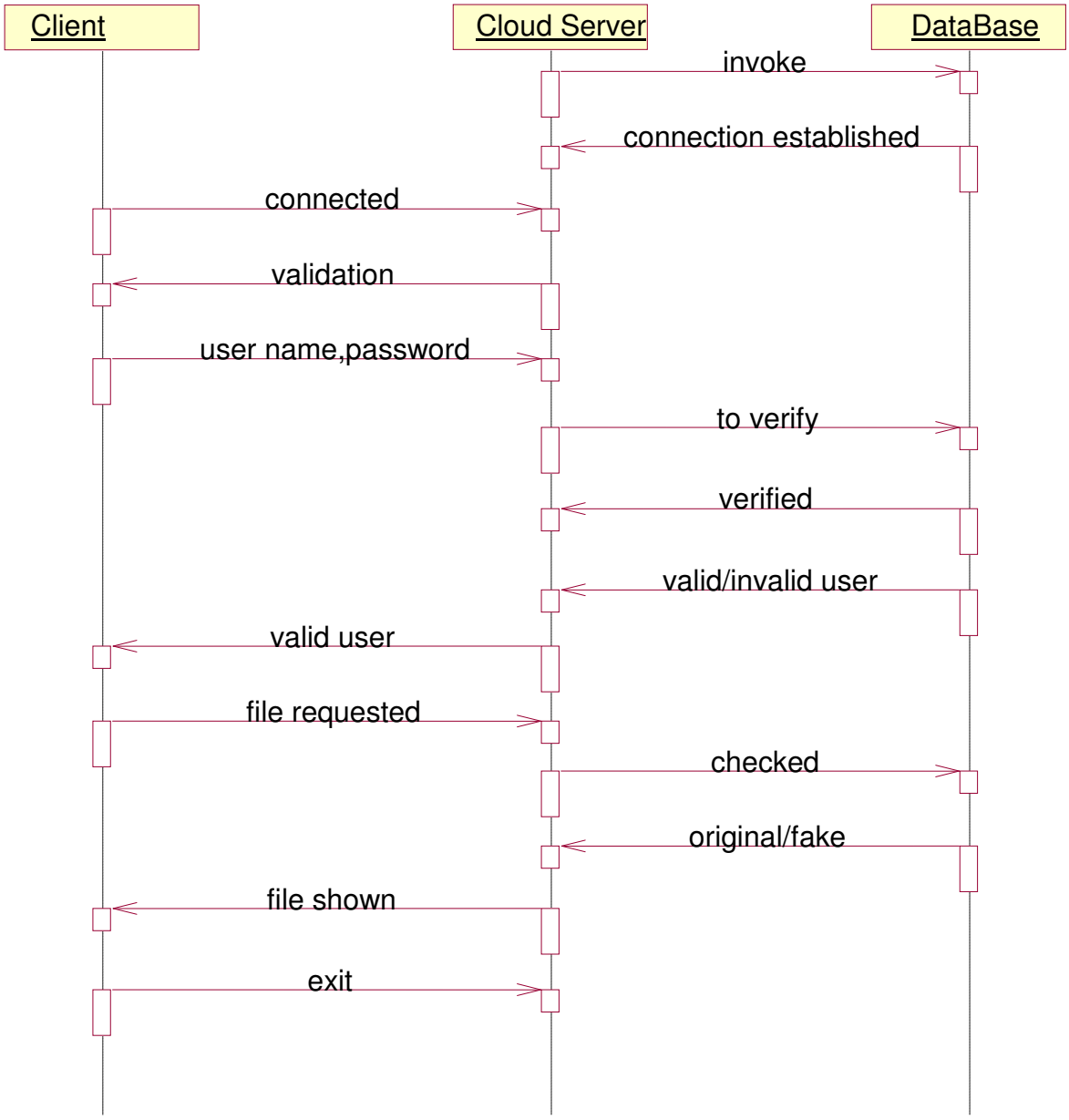

Fig.14 Sequence Diagram

Ref – http://www.akitarescueoftulsa.com/sequence-diagram-dto-data-layer/

Description

As shown, first the client logs into the system and gets connected to the cloud server. The cloud server invokes the database and establishes connection with it. After that the user credentials are checked with the database. After that user can make any request to the server which will be granted after retrieving data from the database.

3.5.4 CLASS DIAGRAM

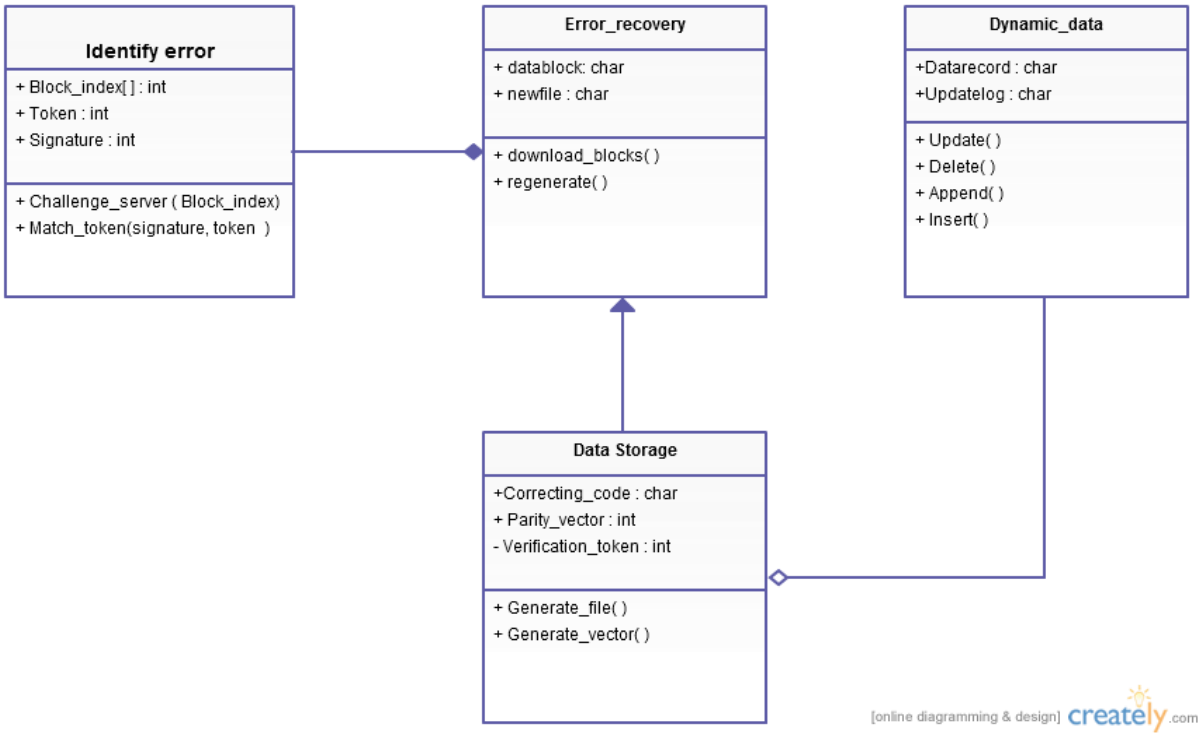

Fig 15 Class Diagrams

Ref – Self made

Description

As shown, we can see that the user can either store the data into the server. Or he can modify the data which will be implanted in the server using the dynamic_data class. This class enables operations like update, delete, append or insert. In case of any error, the system resorts to identify error and eror_recovery classes.

3.5.5 USE CASE DIAGRAM

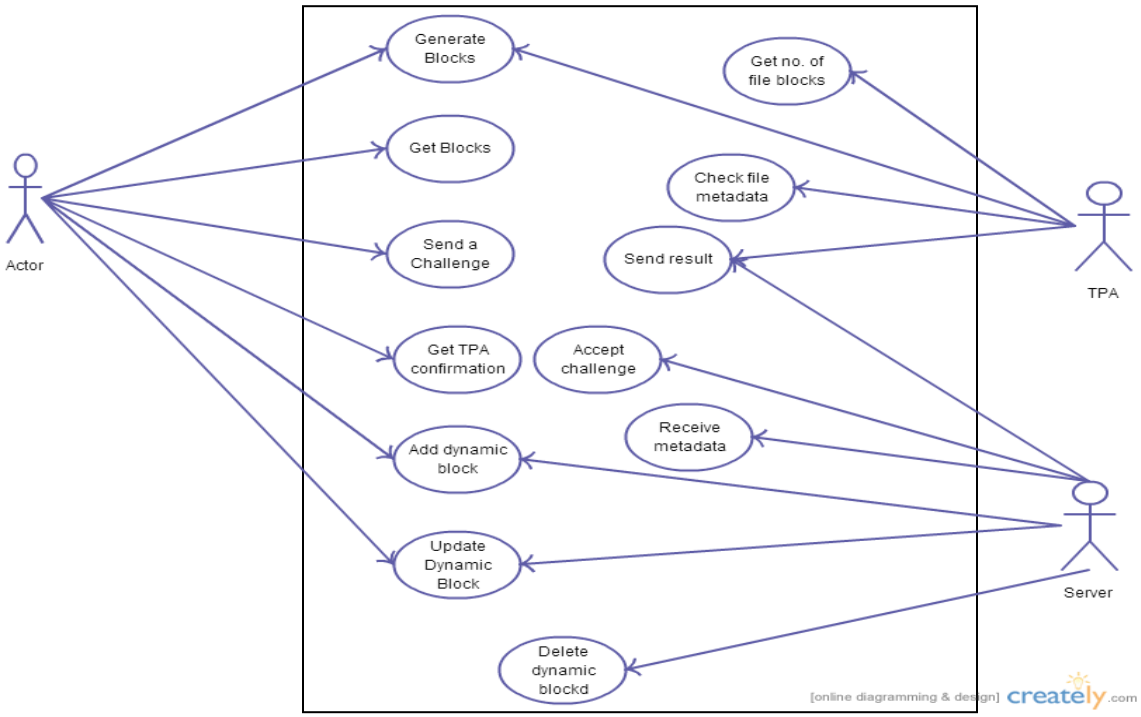

Fig 16 Use case Diagram

Self – made

Description

The user generates data block on the cloud data center. When storage correctness is to be checked, a request is made by the user in form of a 'challenge'. This challenge is accepted by the server. The server receives the metadata i.e. the location and other configuration of the data. The server with the help of TPA checks the correctness and sends the result to the user. This is the basic technique implemented in the security system.

CHAPTER – 4
IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. [11]

The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods. [11]

4.1 Main Modules

4.1.1 Client Module

In this module, the client sends the query to the server. Based on the query the server sends the corresponding file to the client. Before this process, the client authorization step is involved. In the server side, it checks the client name and its password for security process. If it is satisfied, it receives the queries from the client and searches the corresponding files in the database. Finally, finds that file and sends to the client. If the server finds the intruder means, it set the alternative Path to that intruder. [11]

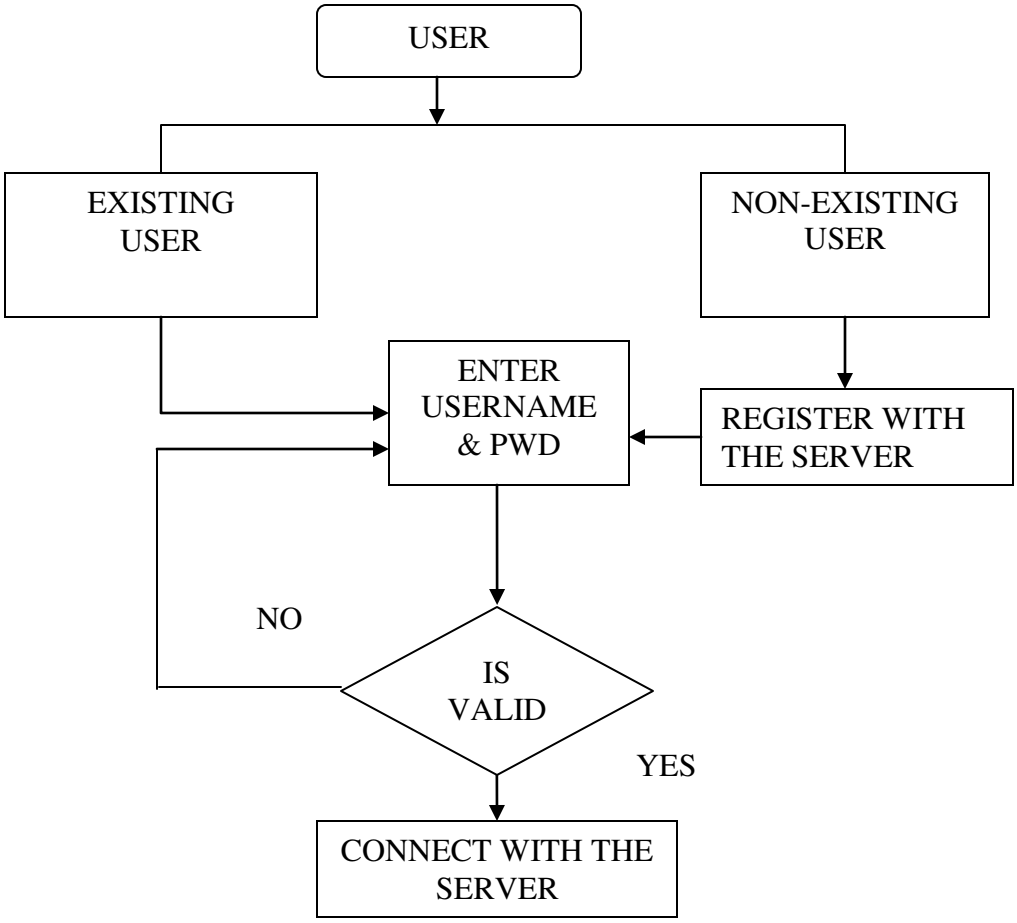

Fig 17 Control Flow

Self –made

4.1.2. System Module

Representative network architecture for cloud data storage is illustrated in Figure 1. Three different network entities can be identified as follows

• **User**

Users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations. [4]

• Cloud Service Provider (CSP)

A CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems. [4]

• Third Party Auditor (TPA):

An optional TPA, who has expertise and capabilities that users may not have, is Trusted to assess and expose risk of cloud storage services on behalf of the users upon request. [4]

4.1.3 Cloud data storage Module

Cloud data storage, a user stores his data through a CSP into a set of cloud servers, which are running in a simultaneous, the user interacts with the cloud servers via CSP to access or retrieve his data. In some cases, the user may need to perform block level operations on his data users should be equipped with security means so that they can make continuous correctness assurance of their stored data even without the existence of local copies. In case that users do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the tasks to an optional trusted TPA of their respective choices. In our model, we assume that the point-to-point communication channels between each cloud server and the user is authenticated and reliable, which can be achieved in practice with little overhead. [12]

4.1.4 Cloud Authentication Server

The Authentication Server (AS) functions as any AS would with a few additional behaviors added to the typical client-authentication protocol. The first addition is the sending of the client authentication information to the masquerading router. The AS in this model also functions as a ticketing authority, controlling permissions on the application network. The other optional function that should be supported by the AS is the updating of client lists, causing a reduction in authentication time or even the removal of the client as a valid client depending upon the request. [12]

4.1.5 Unauthorized data modification and corruption module

One of the key issues is to effectively detect any unauthorized data modification and corruption, possibly due to server compromise and/or random Byzantine failures. Besides, in the distributed case when such inconsistencies are successfully detected, to find which server the data error lies in is also of great significance.

4.1.6 Adversary Module

Security threats faced by cloud data storage can come from two different sources. On the one hand, a CSP can be self-interested, untrusted and possibly malicious. Not only does it desire to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide a data loss incident due to management errors, Byzantine failures and so on.[4]

On the other hand, there may also exist an economically motivated adversary, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete users' data while remaining undetected by CSPs for a certain period. Specifically, we consider two types of adversary with different levels of capability in this paper:

Weak Adversary: The adversary is interested in corrupting the user's data files stored on individual servers. Once a server is comprised, an adversary can pollute the original data files by modifying or introducing its own fraudulent data to prevent the original data from being retrieved by the user. [4]

Strong Adversary: This is the worst case scenario, in which we assume that the adversary can compromise all the storage servers so that he can intentionally modify the data files as long as they are internally consistent. In fact, this is equivalent to the case where all servers are colluding together to hide a data loss or corruption incident. [4]

4.1.7 Sample Screens

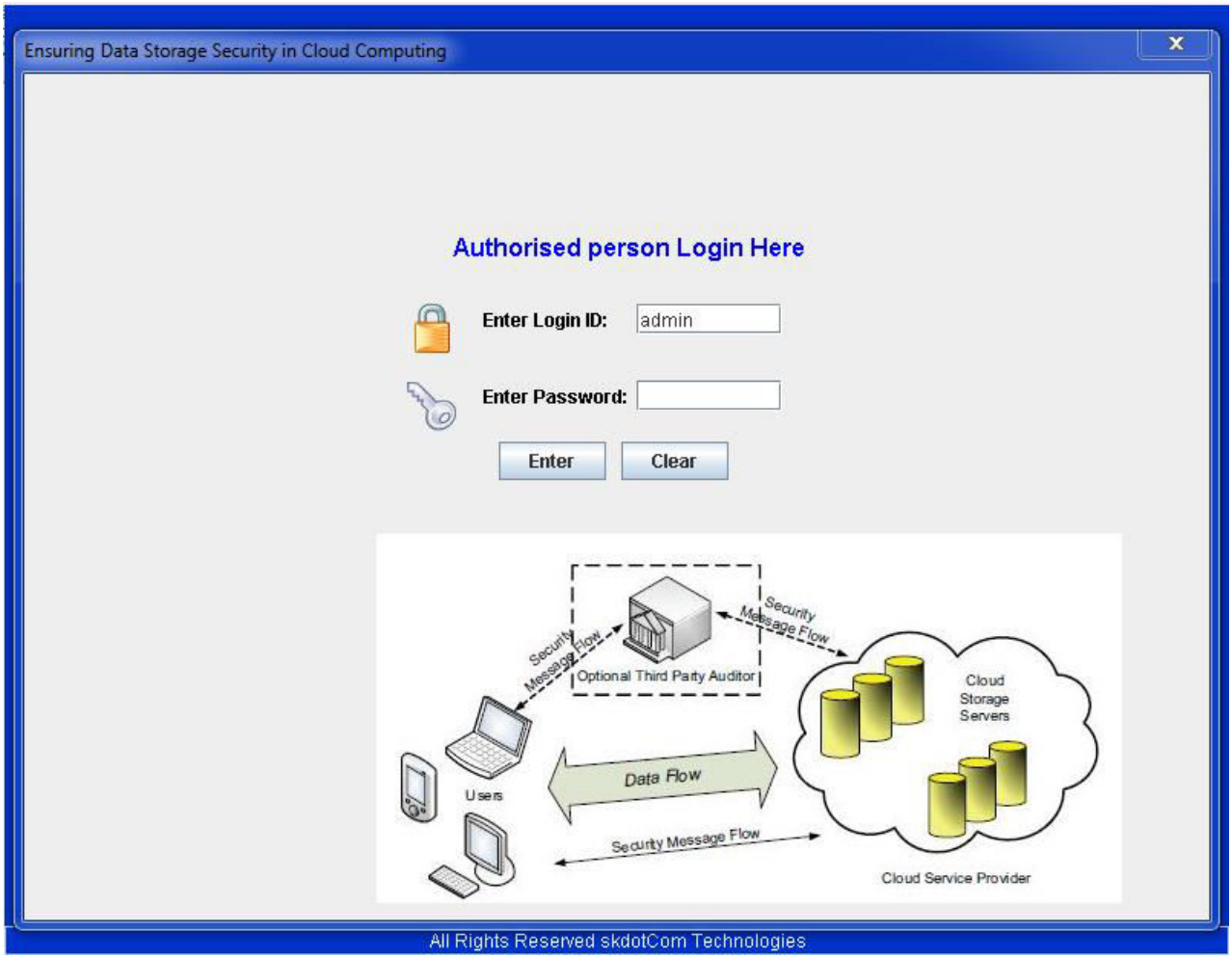

Fig 16 – Admin Login

Ref – Project Output

Fig 17 – Cloud Authentication Server

Ref- Project Output

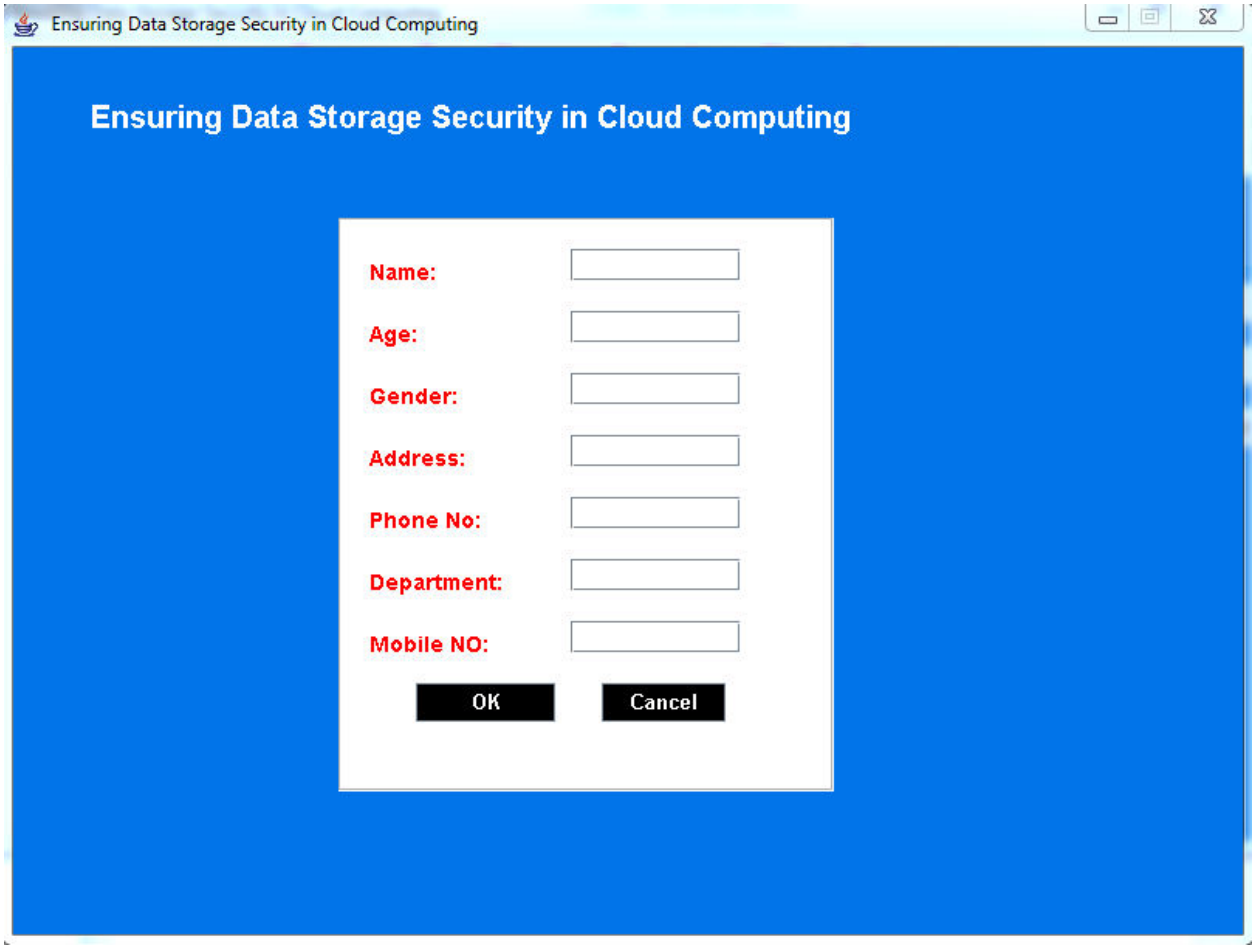

Fig 18 – Add new user

Ref – Project Output

4.2 Database connectivity

4.2.1 Lesson: JDBC Introduction

The JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC helps you to write Java applications that manage these three programming activities: [13]

1. Connect to a data source, like a database
2. Send queries and update statements to the database
3. Retrieve and process the results received from the database in answer to your query [13]

The following simple code fragment gives a simple example of these three steps:

```
public void connectToAndQueryDatabase(String username, String password) {

   Connection con = DriverManager.getConnection(
                    "jdbc:myDriver:myDatabase",
                    username,
                    password);

      Statement stmt = con.createStatement();
   ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");

   while (rs.next()) {
      int x = rs.getInt("a");
      String s = rs.getString("b");
      float f = rs.getFloat("c");
   }
}
```

This short code fragment instantiates a DriverManager object to connect to a database driver and log into the database, instantiates a Statement object that carries your SQL language query to the database; instantiates a ResultSet object that retrieves the results of your query, and executes a simple while loop, which retrieves and displays those results. It's that simple.

4.3 Cloud Simulation

In the past decade, Grids have evolved as the infrastructure for delivering high-performance services for compute and data-intensive scientific applications. To support research and development of new Grid components, policies, and middleware; several Grid simulators, such as GridSim, SimGrid, and GangSim have been proposed. SimGrid is a generic framework for simulation of distributed applications on Grid platforms. Similarly, GangSim is a Grid simulation toolkit that provides support for modeling of Grid-based virtual organisations and resources. On the other hand, GridSim is an event-driven simulation toolkit for heterogeneous Grid resources. It supports modeling of grid entities, users, machines, and network, including network traffic. Although the aforementioned toolkits are capable of modeling and simulating the Grid application behaviors (execution, scheduling, allocation, and monitoring) in a distributed environment consisting of multiple Grid organisations, none of these are able to support the infrastructure and application-level requirements arising from Cloud computing paradigm. In particular, there is very little or no support in existing Grid simulation toolkits for modeling of on-demand virtualization enabled resource and application management. Further, Clouds promise to deliver services on subscription-basis in a pay-as-you-go model to Cloud customers. Hence, Cloud infrastructure modeling and simulation toolkits must provide support for economic entities such as Cloud brokers and Cloud exchange for enabling real-time trading of services between customers and providers. Among the currently available simulators discussed in this paper, only GridSim offers support for economic-driven resource management and application scheduling simulation. Another aspect related to Clouds that should be considered is that research and development in Cloud computing systems, applications and services are in their infancy. There are a number of important issues that need detailed investigation

along the Cloud software stack. Topics of interest to Cloud developers include economic strategies for provisioning of virtualized resources to incoming user's requests, scheduling of applications, resources discovery, inter-cloud negotiations, and federation of clouds. To support and accelerate the research related to Cloud computing systems, applications and services; it is important that the necessary software tools are designed and developed to aid researchers. [14]

4.3.1 CloudSim Architecture

Figure 3 shows the layered implementation of the CloudSim software framework and architectural components. At the lowest layer is the SimJava discrete event simulation engine that implements the core functionalities required for higher-level simulation frameworks such as queuing and processing of events, creation of system components (services, host, data center, broker, virtual machines), communication between components, and management of the simulation clock. Next follows the libraries implementing the GridSim toolkit that support: (i) high level software components for modeling multiple Grid infrastructures, including networks and associated traffic profiles; and (ii) fundamental Grid components such as the resources, data sets, workload traces, and information services. The CloudSim is implemented at the next level by programmatically extending the core functionalities exposed by the GridSim layer. CloudSim provides novel support for modeling and simulation of virtualized Cloudbased data center environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim layer manages the instantiation and execution of core entities (VMs, hosts, data centers, application) during the simulation period. This layer is capable of concurrently instantiating and transparently managing a large scale Cloud infrastructure consisting of thousands of system components. The fundamental issues such as provisioning of hosts to VMs based on user requests, managing application execution, and dynamic monitoring are handled by this layer. A Cloud provider, who wants to study the efficacy of different policies in allocating its hosts, would need to implement his strategies at this layer by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer on how a host is allocated to different competing VMs in the Cloud. A Cloud host can be concurrently

shared among a number of VMs that execute applications based on user-defined QoS specifications. The top-most layer in the simulation stack is the User Code that exposes configuration related functionalities for hosts (number of machines, their specification and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, and broker scheduling policies. A Cloud application developer can generate: (i) a mix of user request distributions, application configurations; and (ii) Cloud availability scenarios at this layer and perform robust tests based on the custom configurations already supported within the CloudSim. [14]

As Cloud computing is a rapidly evolving research area, there is a severe lack of defined standards, tools and methods that can efficiently tackle the infrastructure and application level complexities. Hence in the near future there would be a number of research efforts both in academia and industry towards defining core algorithms, policies, application benchmarking based on execution contexts. By extending the basic functionalities already exposed by CloudSim, researchers would be able to perform tests based on specific scenarios and configurations, hence allowing the development of best practices in all the critical aspects related to Cloud Computing. [14]

One of the design decisions that we had to make as the CloudSim was being developed was whether to extensively reuse existing simulation libraries and frameworks or not.

We decided to take advantage of already implemented and proven libraries such as GridSim and SimJava to handle low-level requirements of the system. For example, by using SimJava, we avoided reimplementation of event handling and message passing among components. This saved us time and cost of software engineering and testing. Similarly, the use of the GridSim framework allowed us to reuse its implementation of networking, information services, files, users, and resources. Since,SimJava and GridSim have been extensively utilized in conducting cutting edge research in Grid resource management by several researchers. Therefore, bugs that may compromise the validity of the simulation have been already detected and fixed. By reusing these long validated frameworks, we were able to focus on critical aspects of the system that are relevant to Cloud computing. At the same time taking advantage of the reliability of components that are not directly related to Clouds. [14]

4.3.2 Modeling the Cloud

The core hardware infrastructure services related to the Clouds are modeled in the simulator by a Datacenter component for handling service requests. These requests are application elements sandboxed within VMs, which need to be allocated a share of processing power on Datacenter's host components. By VM processing, we mean a set of operations related to VM life cycle: provisioning of a host to a VM, VM creation, VM destruction, and VM migration. [14]

A Datacenter is composed by a set of hosts, which are responsible for managing VMs during their life cycles. Host is a component that represents a physical computing node in a Cloud: it is assigned a pre-configured processing capability (expressed in millions of instructions per second – MIPS), memory, storage, and a scheduling policy for allocating processing cores to virtual machines. The Host component implements interfaces that support modeling and simulation of both single-core and multi-core nodes. Allocation of application-specific VMs to Hosts in a Cloud-based data center is the responsibility of the Virtual Machine Provisioner component. This component exposes a number of custom methods for researchers, which aids in implementation of new VM provisioning policies based on optimization goals (user centric, system centric). The default policy implemented by the VM Provisioner is a straightforward policy that allocates a VM to the Host in First-Come-First-Serve (FCFS) basis. The system parameters such as the required number of processing cores, memory and storage as requested by the Cloud user form the basis for such mappings. Other complicated policies can be written by the researchers based on the infrastructure and application demands. For each Host component, the allocation of processing cores to VMs is done based on a host allocation. The policy takes into account how many processing cores will be delegated to each VM, and how much of the processing core's capacity will effectively be attributed for a given VM. So, it is possible to assign specific CPU cores to specific VMs (a space-shared policy) or to dynamically distribute the capacity of a core among VMs (time-shared policy), and to assign cores to VMs on demand, or to specify other policies. [14]

Each Host component instantiates a VM scheduler component that implements the space-shared or timeshared policies for allocating cores to VMs. Cloud system developers and

researchers can extend the VM scheduler component for experimenting with more custom allocation policies. Next, the finer level details related to the timeshared and space-shared policies are described. [14]

4.3.3 Modeling the VM allocation

One of the key aspects that make a Cloud computing infrastructure different from a Grid computing is the massive deployment of virtualization technologies and tools. Hence, as compared to Grids, we have in Clouds an extra layer (the virtualization) that acts as an execution and hosting environment for Cloud-based application services. [14]

Hence, traditional application mapping models that assign individual application elements to computing nodes do not accurately represent the computational abstraction which is commonly associated with the Clouds. For example, consider a physical data center host that has single processing core, and there is a requirement of concurrently instantiating two VMs on that core. Even though in practice there is isolation between behaviors (application execution context) of both VMs, the amount of resources available to each VM is constrained by the total processing power of the host. This critical factor must be considered during the allocation process, to avoid creation of a VM that demands more processing power than the one available in the host, as multiple task units in each virtual machine shares time slices of the same processing core. [14]

To allow simulation of different policies under varying levels of performance isolation, CloudSim supports VM scheduling at two levels: First, at the host level and second, at the VM level. At the host level, it is possible to specify how much of the overall processing power of each core in a host will be assigned to each VM. At the VM level, the VMs assign specific amount of the available processing power to the individual task units that are hosted within its execution engine. [14]

At each level, CloudSim implements the time-shared and space-shared resource allocation

4.3.4 Design and Implementation of CloudSim

In this section, there are finer details related to the fundamental classes of CloudSim, which are building blocks of the simulator.

4.3.4.1 DataCenter This class models the core infrastructure level services (hardware, software) offered by resource providers in a Cloud computing environment. It encapsulates a set of compute hosts that can be either homogeneous or heterogeneous as regards to their resource configurations (memory, cores, capacity, and storage). Furthermore, every DataCenter component instantiates a generalized resource provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices. [14]

4.3.4.2 DatacenterBroker This class models a broker, which is responsible for mediating between users and service providers depending on users' QoS requirements and deploys service tasks across Clouds. The broker acting on behalf of users identifies suitable Cloud service providers through the Cloud Information Service (CIS) and negotiates with them for an allocation of resources that meet QoS needs of users. The researchers and system developers must extend this class for conducting experiments with their custom developed application placement policies. [14]

4.3.4.3 SANStorage. This class models a storage area network that is commonly available to Cloud-based data centers for storing large chunks of data. SANStorage implements a simple interface that can be used to simulate storage and retrieval of any amount of data, at any time subject to the availability of network bandwidth. Accessing files in a SAN at run time incurs additional delays for task unit execution, due to time elapsed for transferring the required data files through the data center internal network. [14]

4.3.4.4 VirtualMachine. This class models an instance of a VM, whose management during its life cycle is the responsibility of the Host component. As discussed earlier, a host can simultaneously instantiate multiple VMs and allocate cores

based on predefined processor sharing policies (space-shared, time-shared). Every VM component has access to a component that stores the characteristics related to a VM, such as memory, processor, storage, and the VM's internal scheduling policy, which is extended from the abstract component called VMScheduling. [14]

4.3.4.5 Cloudlet. This class models the Cloud-based application services (content delivery, social networking, business workflow), which are commonly deployed in the data centers. CloudSim represents the complexity of an application in terms of its computational requirements. Every application component has a pre-assigned instruction length (inherited from GridSim's Gridlet component) and amount of data transfer (both pre and post fetches) that needs to be undertaken for successfully hosting the application. [14]

4.3.4.6 Cloud Coordinator. This abstract class provides federation capacity to a data center. This class is responsible for not only communicating with other peer CloudCoordinator services and Cloud Brokers (DataCenterBroker), but also for monitoring the internal state of a data center that plays integral role in loadbalancing/ application scaling decision making. The monitoring occurs periodically in terms of simulation time. [14] The specific event that triggers the load migration is implemented by CloudSim users through Sensor component. Each sensor may model one specific triggering procedure that may cause the CloudCoordinator to undertake dynamic load-shredding. [14]

4.3.4.7 BWProvisioner. This is an abstract class that models the provisioning policy of bandwidth to VMs that are deployed on a Host component. The function of this component is to undertake the allocation of network bandwidths to set of competing VMs deployed across the data center. Cloud system developers and researchers can extend this class with their own policies (priority, QoS) to reflect the needs of their applications. [14]

4.3.4.8 MemoryProvisioner. This is an abstract class that represents the provisioning policy for allocating memory to VMs. This component models policies for

allocating physical memory spaces to the competing VMs. The execution and deployment of VM on a host is feasible only if the MemoryProvisioner component determines that the host has the amount of free memory, which is requested for the new VM deployment. [14]

4.3.4.9 VMProvisioner. This abstract class represents the provisioning policy that a VM Monitor utilizes for allocating VMs to Hosts. The chief functionality of the VMProvisioner is to select available host in a data center, which meets the memory, storage, and availability requirement for a VM deployment. The default SimpleVMProvisioner implementation provided with the CloudSim package allocates VMs to the first available Host that meets the aforementioned requirements. Hosts are considered for mapping in a sequential order. However, more complicated policies can be easily implemented within this component for achieving optimized allocations, for example, selection of hosts based on their ability to meet QoS requirements such as response time, budget. [14]

4.3.4.10 VMMAllocationPolicy. This is an abstract class implemented by a Host component that models the policies (space-shared, time-shared) required for allocating processing power to VMs. The functionalities of this class can easily be overridden to accommodate application specific processor sharing policies. [14]

4.4 Entities and threading

As the CloudSim programmatically builds upon the SimJava discrete event simulation engine, it preserves the SimJava's threading model for creation of simulation entities. A programming component is referred to as an entity if it directly extends the core Sim_Entity component of SimJava, which implements the Runnable interface. Every entity is capable of sending and receiving messages through the SimJava's shared event queue. The message propagation (sending and receiving) occurs through input and output ports that SimJava associates with each entity in the simulation system. Since threads incur a lot of memory and processor context switching overhead, having a large number

of threads/entities in a simulation environment can be performance bottleneck due to limited scalability. To counter this behavior, CloudSim minimizes the number of entities in the system by implementing only the core components (Users and Datacenters) as the inherite members of SimJava entities. This design decision is significant as it helps CloudSim in modeling a really large scale simulation environment on a computing machine (desktops, laptops) with moderate processing capacity. [14]

Other key CloudSim components such as VMs, provisioning policies, hosts are instantiated as standalone objects, which are lightweight and do not compete for processing power. Hence, regardless of the number of hosts in a simulated data center, the runtime environment (Java virtual machine) needs to manage only two threads (Datacenter and Broker). As the processing of task units is handled by respective VMs, therefore their (task) progress must be updated and monitored after every simulation step. To handle this, an internal event is generated regarding the expected completion time of a task unit to inform the Datacenter entity about the future completion events. Thus, at each simulation step, each Datacenter invokes a method called updateVMsProcessing() for every host in the system, to update processing of tasks running within the VMs. The argument of this method is the current simulation time and the return type is the next expected completion time of a task running in one of the VMs on a particular host. The least time among all the finish times returned by the hosts is noted for the next internal event. [14]

At the host level, invocation of updateVMsProcessing() triggers an updateGridletsProcessing() method, which directs every VM to update its tasks unit status (finish, suspended, executing) with the Datacenter entity. This method implements the similar logic as described previously for updateVMsProcessing() but at the VM level. Once this method is called, VMs return the next expected completion time of the task units currently managed by them. The least completion time among all the computed values is send to the Datacenter entity. As a result, completion times are kept in a queue that is queried by Datacenter after each event processing step. If there are completed tasks waiting in the queue, then they are removed from it and sent back to the user. [14]

4.4.1 Communication among Entities

In the beginning of the simulation, each Datacenter entity registers itself with the CIS (Cloud Information Service) Registry. CIS provides database level match-making services for mapping user requests to suitable Cloud providers. Brokers acting on behalf of users consult the CIS service about the list of Clouds who offer infrastructure services matching user's application requirements. In case the match occurs the broker deploys the application with the Cloud that was suggested by the Cloud management is not a standalone entity. For private or hybrid clouds to be successful, they need to integrate and work with your existing IT management infrastructure and best practices. There has been a lot of debate about what capabilities constitute a private or hybrid cloud deployment. Regardless of whose definition you use, automating the delivery of IT services requires orchestrating service deployment across multiple technology components, such as configuring physical servers, storage, networks, virtualization hypervisors, connection brokers, and so on. Unless you have a very basic infrastructure, delivering IT services requires integrating with the surrounding management ecosystem. This can require generating or acknowledging a work order ticket, updating a Configuration Management Database (CMDB), or installing software using existing server automation tools. Most cloud automation products only automate a portion of your provisioning, ongoing management, or decommissioning processes. To facilitate integration with current IT infrastructure and management ecosystems, a cloud management platform must deliver broad multivendor support as well as extensible architecture. When evaluating cloud management platforms, you need to assess whether a solution has the capabilities that enable it to work with your IT infrastructure. Your cloud management choice not only impacts prior investments, it can limit your future investment options. [14]

The Importance of Cloud Management Extensibility According to Wikipedia, "In systems architecture, extensibility means the system is designed to include hooks and mechanisms for expanding and enhancing the capabilities without having to make major changes to the system infrastructure." Often this means making modifications at runtime without requiring changes to the original source code. While extensibility may not be part of an initial cloud production pilot, knowing this option is available can help grow and

evolve your deployment to meet the unique needs of business groups within your enterprise. [14][15]

Let's look at machine provisioning methodology to see how the differing needs of each group affects cloud infrastructure services. • Dev/Test. The Dev/Test group needs machines quickly to test new features. While machines typically are not needed for long periods of time, developers and test engineers want them as soon as they are needed. For their needs, cloning an existing machine or template can be the most appropriate provisioning methodology. • Production. Production machines stay around for long periods of time. For this group, compliance with supported software revision levels and patch management offered by enterprise management tools from BMC, CA, Microsoft, HP, IBM and others are critical. Therefore, the cloud management platform needs to orchestrate the provisioning and management of a production machine using one of these tool sets. • Desktop. Due to the relatively high cost of datacenter storage as compared to desktop storage, desktop administrators look for space-efficient mechanisms for deploying or streaming operating system and application software to their virtual desktops. For this group, provisioning with space-efficient tools such NetApp Flex Clones, Citrix Provisioning Server, or Microsoft App-V is essential to achieving their price point. One company wit h three different groups, each with very different methods for provisioning new systems, is a very common occurrence. The differences between how these groups operate typically are not limited to just provisioning methodologies. For example, approvals, machine naming, network configuration, resource allocation, service levels, resource location, and the management functions users can perform after machines are provisioned, are just a few of the hundreds of attributes that can differentiate a service provided to one business verses another.[14][15]

4.5 Stored Procedures

A stored procedure is a group of SQL statements that can be called by name. In other words, it is executable code, a mini-program that performs a particular task that can be invoked the same way one can call a function or method. Traditionally, stored procedures have been written in a DBMS-specific programming language.

The following code is an example of how to create a very simple stored procedure using the Java programming language. Note that the stored procedure is just a static Java method that contains normal JDBC code. It accepts two input parameters and uses them to change an employee's car number.[14][15]

```
import java.sql.*;
public class UpdateCar {
   public static void UpdateCarNum(int carNo, int empNo)
      throws SQLException {

      Connection con = null;
      PreparedStatement pstmt = null;

      try {
         con = DriverManager.getConnection(
                  "jdbc:default:connection");

         pstmt = con.prepareStatement(
                   "UPDATE EMPLOYEES " +
                   "SET CAR_NUMBER = ? " +
                   "WHERE EMPLOYEE_NUMBER = ?");

         pstmt.setInt(1, carNo);
         pstmt.setInt(2, empNo);
         pstmt.executeUpdate();
      }
      finally {
         if (pstmt != null) pstmt.close();
      }
   }
}
```

1.4.2 Algorithm[4]

Algorithm 1 Token Pre-computation

procedure
  Choose parameters l, n and function f, ϕ;
  Choose the number t of tokens;
  Choose the number r of indices per verification;
  Generate master key K_{prp} and challenge k_{chal};
  for vector $G^{(j)}$, $j \leftarrow 1, n$ **do**
    for round $i \leftarrow 1, t$ **do**
      Derive $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP}.
      Compute $v_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)]$
    end for
  end for
  Store all the v_is locally.
end procedure

Algorithm 2 Correctness Verification and Error Localization

procedure CHALLENGE(i)
  Recompute $\alpha_i = f_{k_{chal}}(i)$ and $k_{prp}^{(i)}$ from K_{PRP};
  Send $\{\alpha_i, k_{prp}^{(i)}\}$ to all the cloud servers;
  Receive from servers:
  $\{R_i^{(j)} = \sum_{q=1}^{r} \alpha_i^q * G^{(j)}[\phi_{k_{prp}^{(i)}}(q)] | 1 \le j \le n\}$
  for ($j \leftarrow m+1, n$) **do**
    $R^{(j)} \leftarrow R^{(j)} - \sum_{q=1}^{r} f_{k_j}(s_{I_q,j}) \cdot \alpha_i^q, I_q = \phi_{k_{prp}^{(i)}}(q)$
  end for
  if $((R_i^{(1)}, \ldots, R_i^{(m)}) \cdot \mathbf{P} == (R_i^{(m+1)}, \ldots, R_i^{(n)}))$ **then**
    Accept and ready for the next challenge.
  else
    for ($j \leftarrow 1, n$) **do**
      if ($R_i^{(j)} != v_i^{(j)}$) **then**
        return server j is misbehaving.
      end if
    end for
  end if
end procedure

Algorithm 3 Error Recovery

1: **procedure**
% Assume the block corruptions have been detected among
% the specified r rows;
% Assume $s \leq k$ servers have been identified misbehaving
2: Download r rows of blocks from servers;
3: Treat s servers as erasures and recover the blocks.
4: Resend the recovered blocks to corresponding servers.
5: **end procedure**

CHAPTER – 5
CONCLUSION

In this paper, the problem of data security in cloud data storage is targeted, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server. Through detailed security and performance analysis, we show that our scheme is highly efficient and resilient to data breaches or other security threats.

Data storage security in Cloud Computing, an area full of challenges and of paramount importance, is still in its initial stage now, and many research problems are yet to be identified. Several possible directions for future research on this area are discovered. The most promising one is a model in which public verifiability is enforced. Public verifiability, supported in allows TPA to audit the cloud data storage without demanding users' time, feasibility or resources. An interesting question in this model is if a scheme can be constructed to achieve both public verifiability and storage correctness assurance of dynamic data. Besides, along with my research on dynamic cloud data storage, I also plan to investigate the problem of fine-grained data error localization.

BIBLIOGRAPHY

1. Office of Privacy Commissioner of Canada. “Introduction to Cloud Computing” Fact Sheet, pp. 1-6. September 2014

2. Wikipedia. September 2014. “Cloud Computing Security” Section 1. [Online] Available: en.wikipedia.org.

3. Ted Samson. 2013, Feb 25. 9 Top threats to Cloud Computing Security. [Online]. Available: http://www.infoworld.com.

4. Cong Wang, Qian Wang, Kui Ren. “Ensuring data storage security in cloud computing." In *INFOCOM, 2010 Proceedings IEEE*, pp. 1-9. Ieee, 2010.

5. Velavan, P., K. Balaji, and R. Ganesh. "Secret keys and the Packets Transportation for Privacy Data Forwarding Method In Cloud Server."2013

6. Sun Microsystems. 2014, September. The Java Technology Phenomenon. [Online] Available: http://www.math.uni-hamburg.de/doc/java/tutorial/getStarted/intro

7. M. Himagireshwar Rao, G. Kiran Kumar. “Risk-Aware Mitigation for MANET Routing Attacks” International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issue 9, pp.3224-3227 September 2014

8. Torry Harris Business Solution. 2015 February. Cloud Computing Overview. [Online] Available: http://www.thbs.com/knowledge-zone/cloud-computing-overview

9. Pearson, Siani, and Azzedine Benameur. "Privacy, security and trust issues arising from cloud computing." In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 693-702. IEEE, 2010.

10. Güçlükol, Alican, Anıl Paçacı, Meriç Taze, and Serbay Arslanhan. "Software Design Description for Web Based Integrated Development Environment DEVCLOUD Web Based Integrated Development Environment."2013

11. Prasanna, K. A. V. L. "PERFORMANCE EVALUATION OF MULTIVIEWPOINT-BASED SIMILARITY MEASURE FOR DATA CLUSTERING." *Journal of Global Research in Computer Science* 3, no. 11 (2012).

12. Biruntha, S., V. Venkatesa Kumar, and S. Palaniswami. "Enabling data storage security in cloud computing for banking enterprise." *Recent Advances in Networking, VLSI and Signal Processing* (2010).

13. Oracle Java™ Documentation. 2014, April. JDBC Introduction. [Online] Available:www.docs.oracle.com/javase/tutorial/jdbc/overview

14. Calheiros, Rodrigo N., Rajiv Ranjan, César AF De Rose, and Rajkumar Buyya. "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services." *pp.:0903.2525* (2009).

15. Buyya, Rajkumar, Rajiv Ranjan, and Rodrigo N. Calheiros. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities." In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*, pp. 1-11. IEEE, 2009.