A Project Report

on

A LIGHTWEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING

Submitted to

Acharya Nagarjuna University
In Partial Fulfilment of the Requirement for the Award of

Bachelor’s Degree in
Information Technology

by

D. V. Vyshnavi	**Y16AIT424**
Ch. Mounika	**Y16AIT422**
D. Ramachandra Reddy	**Y16AIT426**
D. Sunil Kumar	**Y15AIT417**

Under the guidance of
Mr. K. Bhaskara Rao, M. Tech

Department of Information Technology
Bapatla Engineering College
Mahatmaji Puram, Bapatla - 522102

2019-2020

Affiliated to

Acharya Nagarjuna University

Bapatla Engineering College
Department of Information Technology
Mahatmaji Puram, Bapatla – 522102

CERTIFICATE

This is certified that the Project entitled

“A LIGHTWEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING”

submitted by

D. V. Vyshnavi	**Y16AIT424**
Ch. Mounika	**Y16AIT422**
D. Ramachandra Reddy	**Y16AIT426**
D. Sunil Kumar	**Y15AIT417**

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Information Technology) at BAPATLA ENGINEERING COLLEGE, Bapatla under the Acharya Nagarjuna University. This work is done during year 2019-2020, under our guidance.

Date: / /

(Asst. Prof. K. Bhaskara Rao)
Project Guide

(Asst. Prof. M. Praveen Kumar)
Project Coordinator

Sig. of HOD

Sig. of External Examiner

Acknowledgements

We are profoundly grateful to **Mr. K. Bhaskara Rao, M. Tech, Asst. Professor** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards **Dr. V. Damodara Naidu, Ph. D(IIT-K), Principal**, Bapatla Engineering College**, Prof. N. Siva Ram Prasad, M. Tech**, **Head of Department** of Information Technology and **Mr. M. Praveen Kumar, M. Tech, Asst. Professor**, Project Coordinator whose invaluable guidance supported us in completing this project.

At last we must express our sincere heartfelt gratitude to all the staff members of Information Technology Department who helped me directly or indirectly during this course of work.

D. V. Vyshnavi	**Y16AIT424**
Ch. Mounika	**Y16AIT422**
D. Ramachandra Reddy	**Y16AIT426**
D. Sunil Kumar	**Y15AIT417**

ABSTRACT

With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in great need for mobile cloud applications. In this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing. It adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. LDSS moves a large portion of the computationally intensive access control tree transformation in CP-ABE from mobile devices to external proxy servers. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program-based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.

Key words: mobile cloud computing, data encryption, access control, user revocation

Contents

Title **Page No**

List of Figures

List of Tables

Chapter 1

Introduction

With the development of cloud computing and the popularity of smart mobile devices, people are gradually getting accustomed to a new era of data sharing model in which the data is stored on the cloud and the mobile devices are used to store/retrieve the data from the cloud.

Typically, mobile devices only have limited storage space and computing power. On the contrary, the cloud has enormous amount of resources. In such a scenario, to achieve the satisfactory performance, it is essential to use the resources provided by the cloud service provider (CSP) to store and share the data.

Nowadays, various cloud mobile applications have been widely used. In these applications, people (data owners) can upload their photos, videos, documents and other files to the cloud and share these data with other people (data users) they like to share. CSPs also provide data management functionality for data owners. Since personal data files are sensitive, data owners are allowed to choose whether to make their data files public or can only be shared with specific data users.

Clearly, data privacy of the personal sensitive data is a big concern for many data owners. The state-of-the-art privilege management/access control mechanisms provided by the CSP are either not sufficient or not very convenient. They cannot meet all the requirements of data owners. First, when people upload their data files onto the cloud, they are leaving the data in a place where is out of their control, and the CSP may spy on user data for its commercial interests and/or other reasons.

Second, people have to send password to each data user if they only want to share the encrypted data with certain users, which is very cumbersome. To simplify the privilege management, the data owner can divide data users into different groups and send password to the groups which they want to share the data.

However, this approach requires fine-grained access control. In both cases, password management is a big issue. Apparently, to solve the above problems, personal sensitive data should be encrypted before uploaded onto the cloud so that the data is secure against the CSP.

However, the data encryption brings new problems. How to provide efficient access control mechanism on ciphertext decryption so that only the authorized users can access the plaintext data is challenging.

In addition, system must offer data owners effective user privilege management capability, so they can grant/revoke data access privileges easily on the data users. There have been substantial researches on the issue of data access control over ciphertext. In these researches, they have the following common assumptions.

First, the CSP is considered honest and curious. Second, all the sensitive data are encrypted before uploaded to the Cloud. Third, user authorization on certain data is achieved through encryption/decryption key distribution. In general, we can divide these approaches into four categories: simple ciphertext access control, hierarchical access control, access control based on fully homomorphic encryption and access control based on attribute-based encryption (ABE).

All these proposals are designed for non-mobile cloud environment. They consume large amount of storage and computation resources, which are not available for mobile devices. As the mobile cloud becomes more and more popular, providing an efficient secure data sharing mechanism in mobile cloud is in urgent need. To address this issue, in this paper, we propose a Lightweight Data Sharing Scheme (LDSS) for mobile cloud computing environment.

The main contributions of LDSS are as follows:

(1) We design an algorithm called LDSS-CP-ABE based on Attribute-Based Encryption (ABE) method to offer efficient access control over ciphertext.

(2) We use proxy servers for encryption and decryption operations. In our approach, computational intensive operations in ABE are conducted on proxy servers, which greatly reduce the computational overhead on client-side mobile devices. Meanwhile, in LDSS-CP-ABE, in order to maintain data privacy, a

version attribute is also added to the access structure. The decryption key format is modified so that it can be sent to the proxy servers in a secure way.

(3) We introduce lazy re-encryption and description field of attributes to reduce the revocation overhead when dealing with the user revocation problem.

(4) Finally, we implement a data sharing prototype framework based on LDSS. The experiments show that LDSS can greatly reduce the overhead on the client side, which only introduces a minimal additional cost on the server side. Such an approach is beneficial to implement a realistic data sharing security scheme on mobile devices. The results also show that LDSS has better performance compared to the existing ABE based access control schemes over ciphertext.

1.1 Objective of the Project

With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in great need for mobile cloud applications. In this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing. It adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. LDSS moves a large portion of the computational intensive access control tree transformation in CP-ABE from mobile devices to external proxy servers. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.

Chapter 2

Literature Survey

2.1 Survey

[Ruixuan Li; et.al. 2017] in this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing. It adopts CP-ABE, an access control technology used in normal cloud environment. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy revocation, which is a thorny issue in program-based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.[1]

[Gentry C, Halevi S; et.al. 2011] Recent news reveal a powerful attacker which breaks data confidentiality by acquiring cryptographic keys, by means of coercion or backdoors in cryptographic software. In this paper, we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the ciphertext blocks. Our evaluation results suggest that Bastion is well-suited for integration in existing systems since it incurs less than 5% overhead compared to existing semantically secure encryption modes.[2]

[Brakerski Z, Vaikuntanathan V; et.al. 2016] Cloud service providers (CSPs) adapt different pricing models for their offered services. Some of the models are suitable for short term requirement while others may be suitable for the cloud service user's (CSU) long term requirement. For example, reservation-based pricing model is appropriate for a CSU's long-term demand for resources. Finding the optimal amount of resources to be reserved in advance. In this paper, we derive some heuristic-based polynomial time algorithms to find some near optimal solution to this problem. We show that the cost for CSU using our approach is comparable to the solution obtained using optimal IPP.[3]

[Qihua Wang, Hongxia Jin; et.al. 2015] As the rapidly growing volume of data are beyond the capabilities of many computing infrastructures, to securely process them on cloud has become a preferred solution which can both utilize the powerful capabilities provided by cloud and protect data privacy. This paper presents an approach to securely decompose a tensor, a mathematical model widely used in data-intensive applications, to a core tensor multiplied with a certain number of truncated orthogonal bases. [4]

[Adam Skillen and Mohammad Mannan; et.al. 2014] As one of the most popular cloud services, data storage has attracted great attention in recent research efforts. Key-value (k-v) stores have emerged as a popular option for storing and querying billions of key-value pairs. Finally, we apply the kBF to a practical problem of implementing a state machine to demonstrate how the kBF can be used as a building block for more complicated software infrastructures.[5]

[Wang W, Li Z, Owens R; et.al. 2014] Data security is a major concern in cloud computing. We propose a novel multi-layered key structure, called Recursively Encrypted Red-black Key tree (RERK), that ensures no key materials will be leaked, yet the client is able to manipulate keys by performing tree operations in collaboration with the servers. We implement our solution on the Amazon EC2. The experimental results show that our solution can efficiently support the deletion of outsourced data in cloud computing.[6]

[Maheshwari U, Vingralek R, Shapiro W] In this paper, we focus on one such important query operation, namely range query. One of the basic security primitives that can be used to evaluate range queries is secure comparison of encrypted integers. Therefore, in this paper, we first propose an efficient method for converting an encrypted integer z into encryptions of the individual bits of z. We then utilize the proposed security primitive to construct a new protocol for secure evaluation of range queries in the cloud computing environment. [7]

Chapter 3
Software Requirements Specification

3.1. Overall Description

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behaviour of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analysing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers. Projects are subject to three sorts of requirements:

- Business requirements describe in business terms *what* must be delivered or accomplished to provide value.
- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time.

There are aspects in the feasibility study portion of the preliminary investigation:

- **Economic Feasibility:** A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

- **Operational Feasibility:** Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So, there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

- **Technical Feasibility**: Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

3.2 External Interface Requirements

User Interface

The user interface of this system is a user-friendly Java Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through Java capabilities.

Software Interfaces

The required software is JAVA1.6.

Operating Environment

Windows XP

3.2.1. Hardware Requirements

• Processor	-	Pentium –IV
• Speed	-	1.1 Ghz
• RAM	-	256 MB (min)
• Hard Disk	-	20 GB
• Key Board	-	Standard Windows Keyboard
• Mouse	-	Two or Three Button Mouse
• Monitor	-	SVGA

3.2.2 Software Requirements

• Operating System	: Windows XP
• Programming Language	: Java

Chapter 4
Requirement Analysis

4.1 Requirements Gathering stage

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

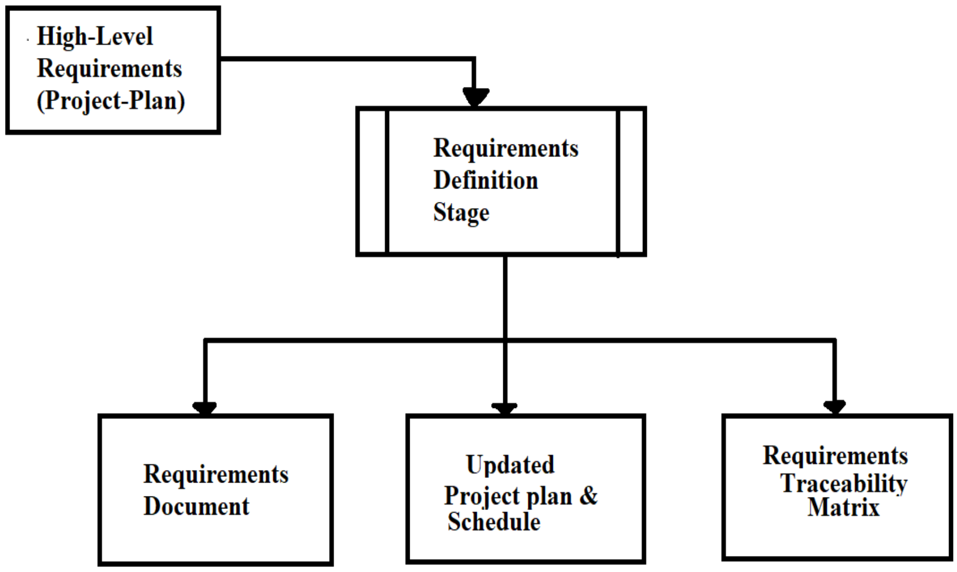

High-Level Requirements (Project-Plan)
Requirements Definition Stage
Requirements Document
Updated Project plan & Schedule
Requirements Traceability Matrix

Figure 4.1: Requirement gathering stage

These requirements are fully described in the primary deliverables for this stage: The Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

4.2 Analysis Stage

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included.

The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high-level estimates of effort for the out stages.

4.3 Software Requirements Analysis

Programming Language: Java

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language
- Java is cohesive and consistent
- Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Finally, Java is to Internet Programming where c was to System Programming.

Importance of Java to the Internet

Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs. in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

- **Applications and applets:** An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Java’s ability to create Applets makes it important. An Applet I san application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet I actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

- **Java Architecture:** Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment. Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.
- **Compilation of code:** When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed t executed the byte code. The JVM is created for the overcoming the issue of probability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.
- **Simple:** Java was designed to be easy for the Professional programmer to learn and to use effectively. If you are an experienced C++ Programmer, learning Java will orient features of C++. Most of the confusing concepts from C++ are either left out of Java or implemented in a cleaner, more approachable manner. In Java there are a small number of clearly defined ways to accomplish a given task.

- **Object oriented:** Java was not designed to be source-code compatible with any other language. This allowed the Java team the freedom to design with a blank state. One outcome of this was a clean usable, pragmatic approach to objects. The object model in Java is simple and easy to extend, while simple types, such as integers, are kept as high-performance non-objects.

- **Robust:** The multi-platform environment of the web places extraordinary demands on a program, because the program must execute reliably in a variety of systems. The ability to create robust programs was given a high priority in the design of Java? Java is strictly typed language; it checks your code at compile time and runtime.

Operating System: Windows XP

- WLAN features. The most valuable enhancement in Windows XP is the way that it intuitively handles 802.11b wireless LAN connectivity.
- System Restore and Device Driver Rollback.
- Remote Desktop and Remote Assistance.
- Application Compatibility Mode.

Chapter 5

System Design

5.1 UML diagrams

The Unified Modelling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

5.1.1 Class diagram

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts:

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

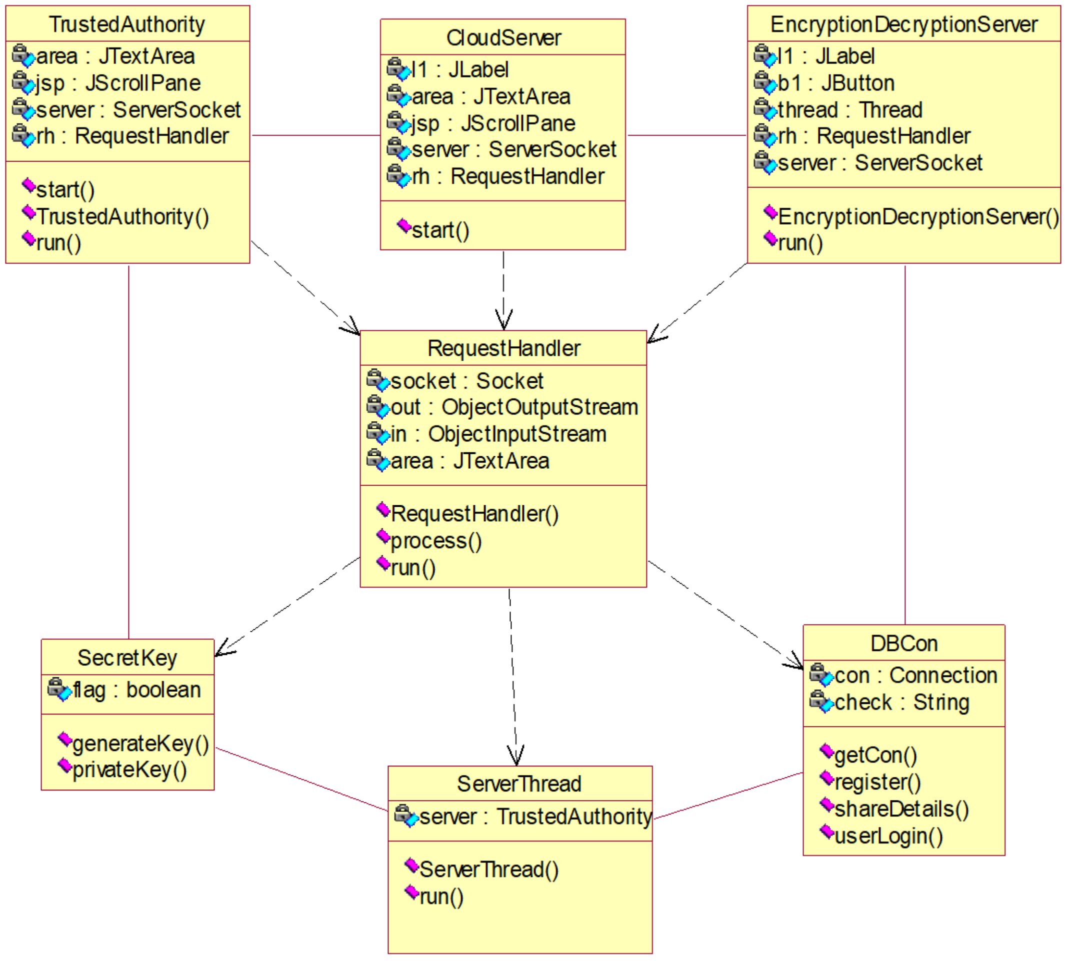


Figure 5.1.1: Class Diagram

5.1.2 Use case diagram

A **use case diagram** at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

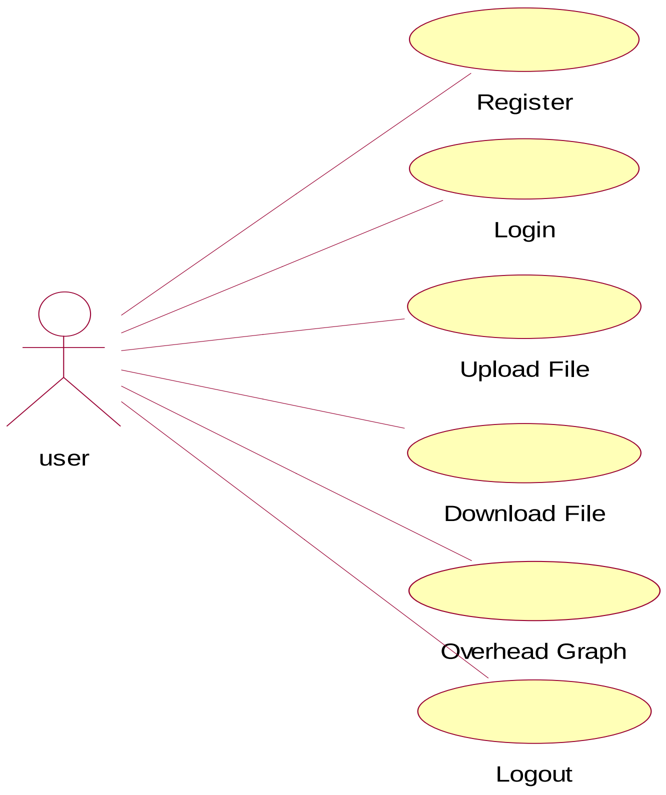


Figure 5.1.2: Use Case Diagram

5.1.3. Sequence Diagram

A **sequence diagram** is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. Sequence diagrams are sometimes called **event diagrams**, **event scenarios**, and timing diagrams.

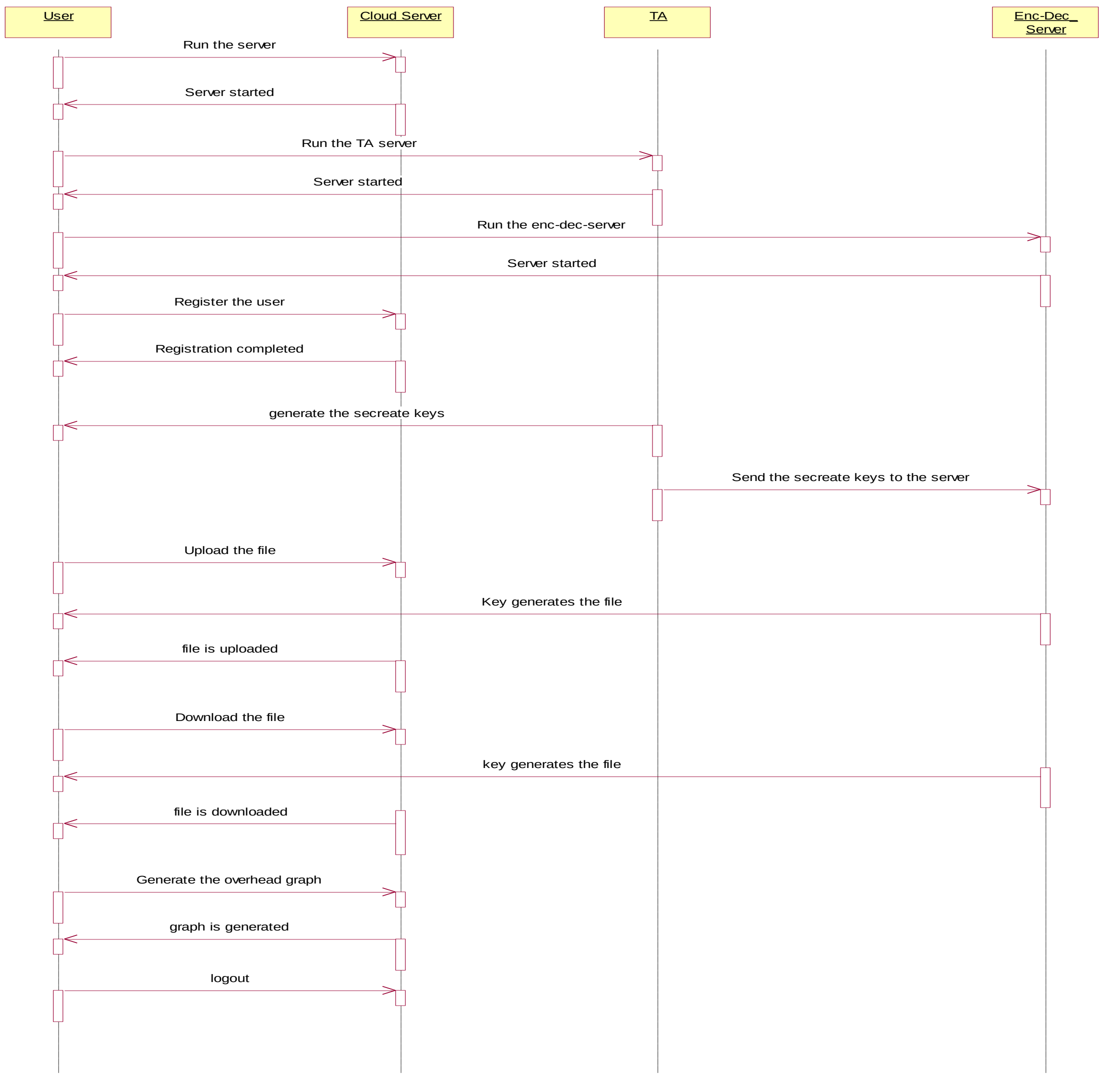


Figure 5.1.3: Sequence Diagram

5.1.4 Collaboration diagram

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.

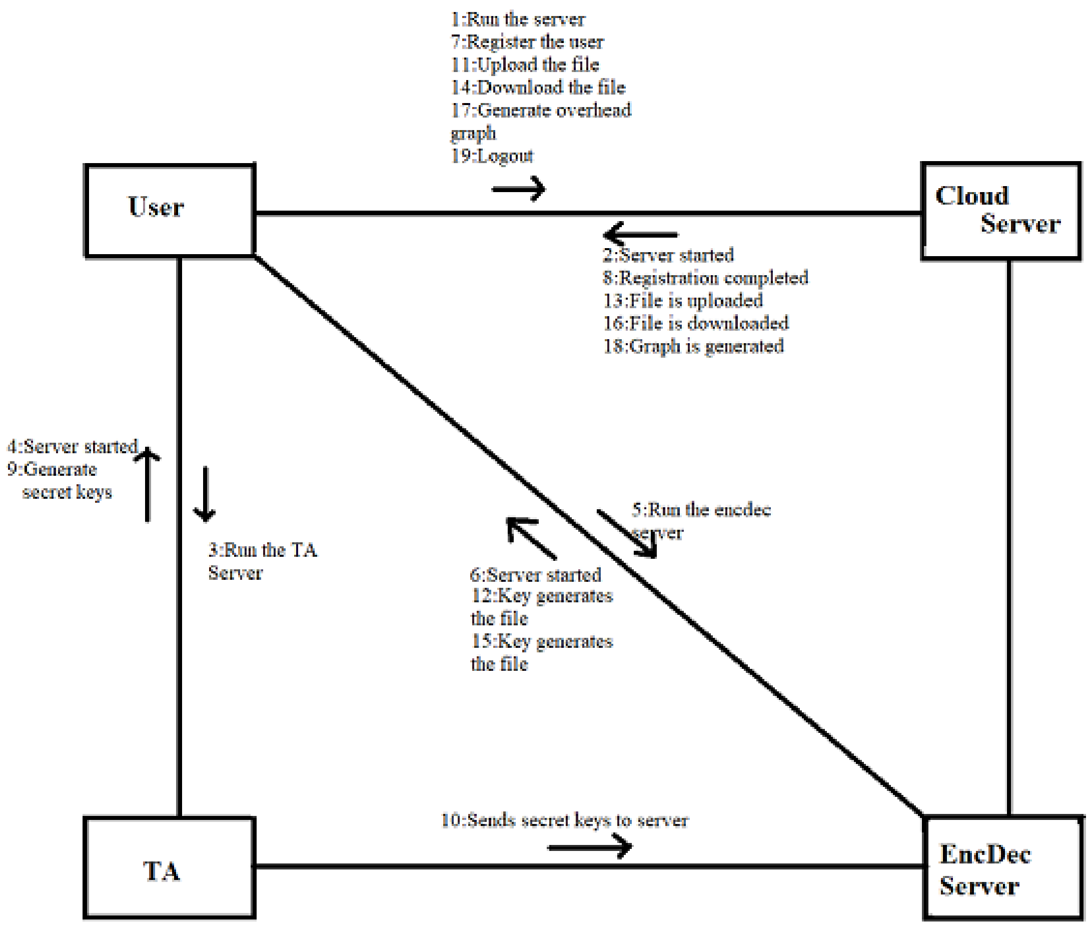


Figure 5.1.4: Collaboration Diagram

5.1.5 Component Diagram

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

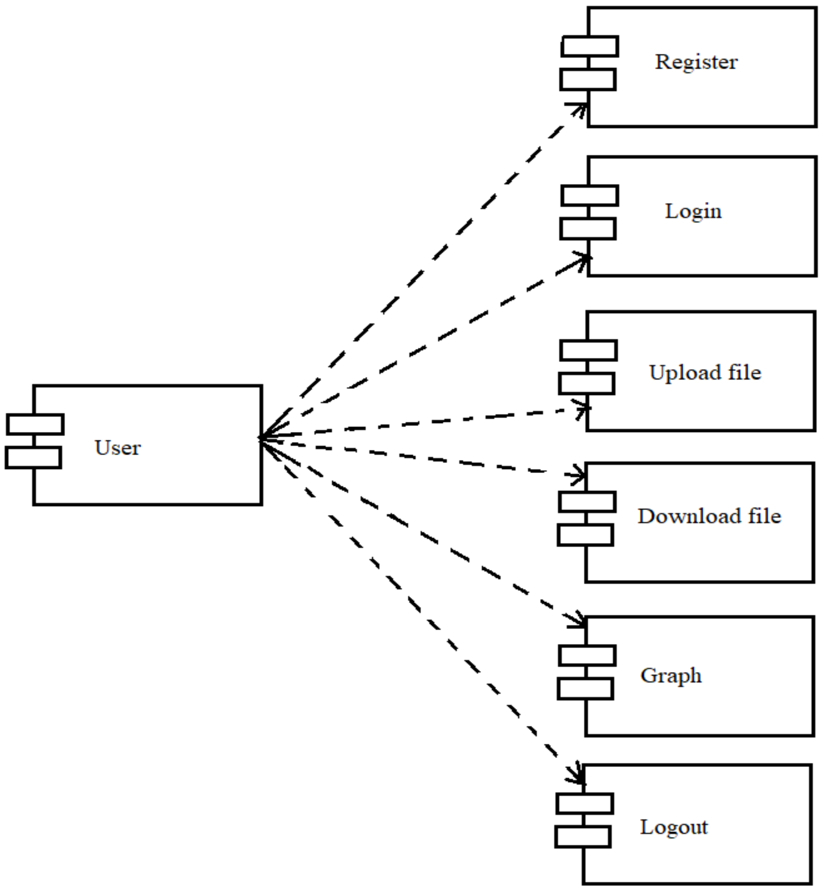


Figure 5.1.5: Component Diagram

5.1.6 Deployment Diagram

A **deployment diagram** in the Unified Modelling Language models the *physical* deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

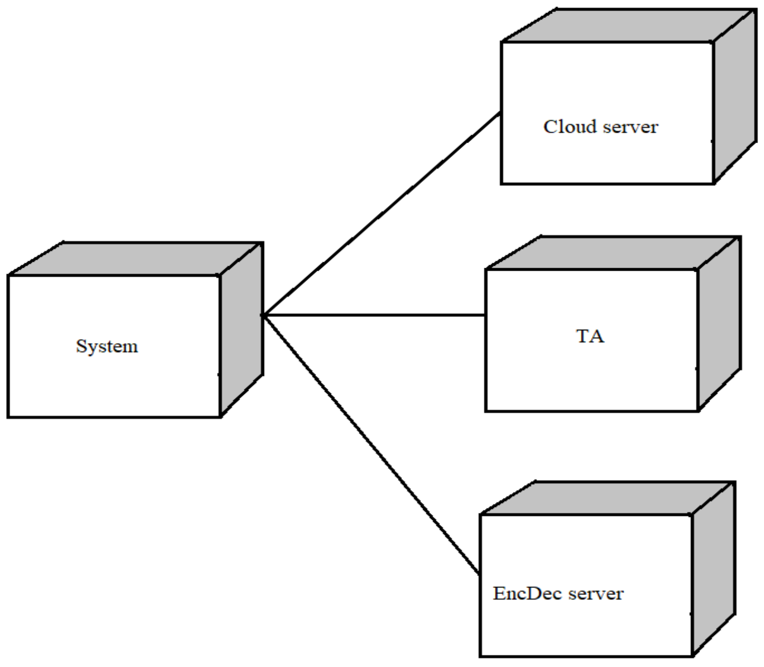


Figure 5.1.6: Deployment Diagram

5.1.7 Activity diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity.

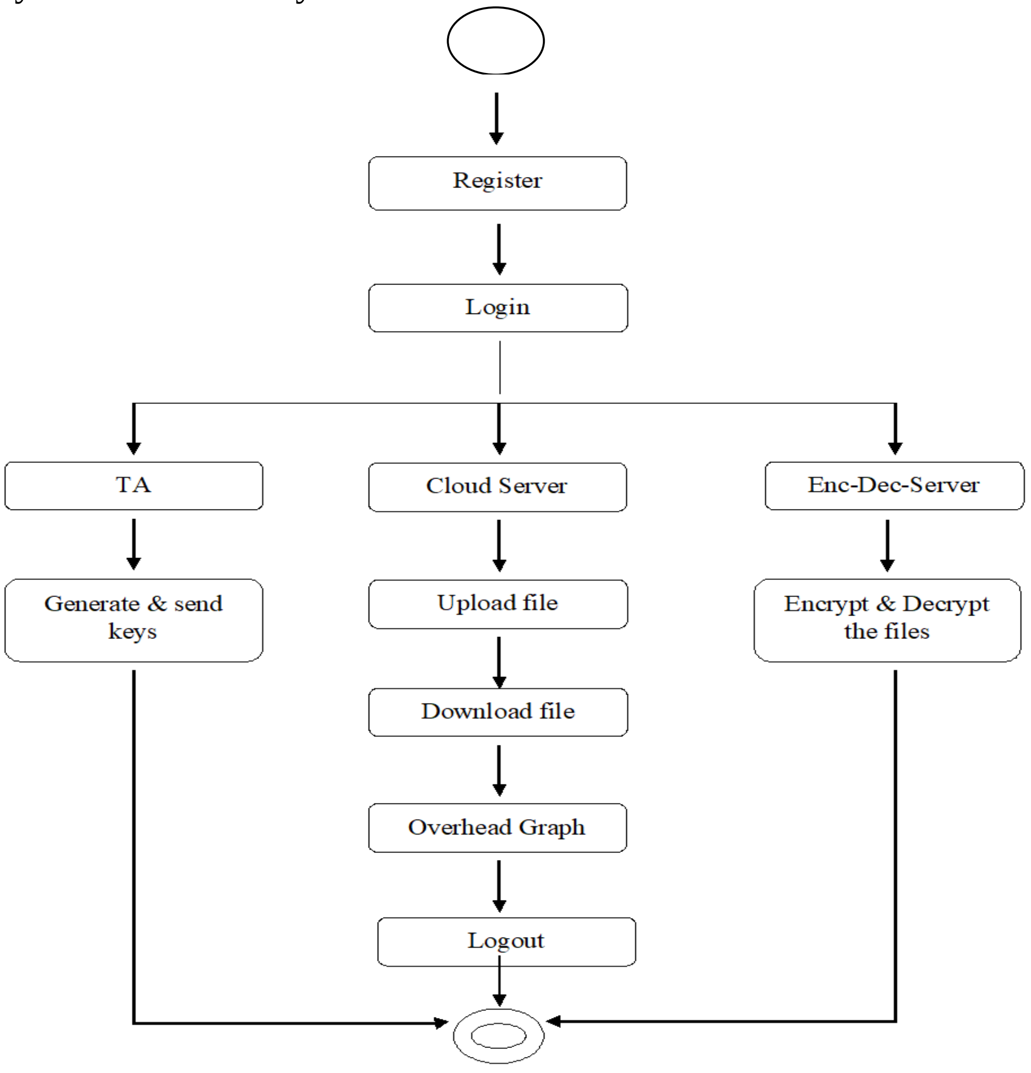


Figure 5.1.7 Activity Diagram

Chapter 6

System Testing

6.1 Testing

Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. Actually, testing is the state of implementation which aimed at ensuring that the system works accurately and efficiently before the actual operation commence.

6.1.1 System Testing

Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus, the code was exhaustively checked for all possible correct data and the outcomes were also checked.

6.2 Test Cases and Test Results

Table 6.2: Test Cases and Test Results

Test Case Id	**Test Case Name**	**Test Case Desc.**	**Test Steps Step**	**Expected**	**Actual**	**Test Case Status**	**Test Priority**
01	Upload file	Verify either file is uploaded or not	If file is not uploaded	We cannot share the data	file is uploaded	High	High
02	Download file	whether the file is download ed or not	If it's not downloaded	We cannot get the files	File is downloade d	High	High
03	Generate the graph	Whether the graph is generated or not	If it's not generated	We cannot get the encryption files	Graph is generated	High	High
04	exit	Whether user is exit or not	If he not exits	He cannot complete the process	He has complete-d the process	High	High

Chapter 7

Project Planning

7.1 Introduction

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process.

7.2 Existing System

Nowadays, various cloud mobile applications have been widely used. In these applications, people (data owners) can upload their photos, videos, documents and other files to the cloud and share these data with other people (data users) they like to share. CSPs also provide data management functionality for data owners. Since personal data files are sensitive, data owners are allowed to choose whether to make their data files public or can only be shared with specific data users. Clearly, data privacy of the personal sensitive data is a big concern for many data owners. The state-of-the-art privilege management/access control mechanisms provided by the CSP are either not sufficient or not very convenient. They cannot meet all the requirements of data owners. First, when people upload their data files onto the cloud, they are leaving the data in a place where is out of their control, and the CSP may spy on user data for its commercial interests and/or other reasons. Second, people have to send password to each data user if they only want to share the encrypted data with certain users, which is very cumbersome. To simplify the privilege management, the data owner can divide data users into different groups and send password to the groups which they want to share the data. However, this approach requires fine-grained access control. In both cases, password management is a big issue.

7.2.1 Disadvantages of Existing System

1. When people upload their data files onto the cloud, they are leaving the data in a place where is out of their control, and the CSP may spy on user data for its commercial interests and/or other reasons.
2. People have to send password to each data user if they only want to share the encrypted data with certain users, which is very cumbersome.

7.3 Proposed System

Personal sensitive data should be encrypted before uploaded onto the cloud so that the data is secure against the CSP. However, the data encryption brings new problems. How to provide efficient access control mechanism on ciphertext decryption so that only the authorized users can access the plaintext data is challenging. In addition, system must offer data owner's effective user privilege management capability. First, the CSP is considered honest and curious. Second, all the sensitive data are encrypted before uploaded to the Cloud. Third, user authorization on certain data is achieved through encryption/decryption key distribution. In general, we can divide these approaches into four categories: simple ciphertext access control, hierarchical access control, access control based on fully homomorphic encryption and access control based on attribute-based encryption (ABE). All these proposals are designed for non-mobile cloud environment. They consume large amount of storage and computation resources, which are not available for mobile devices.

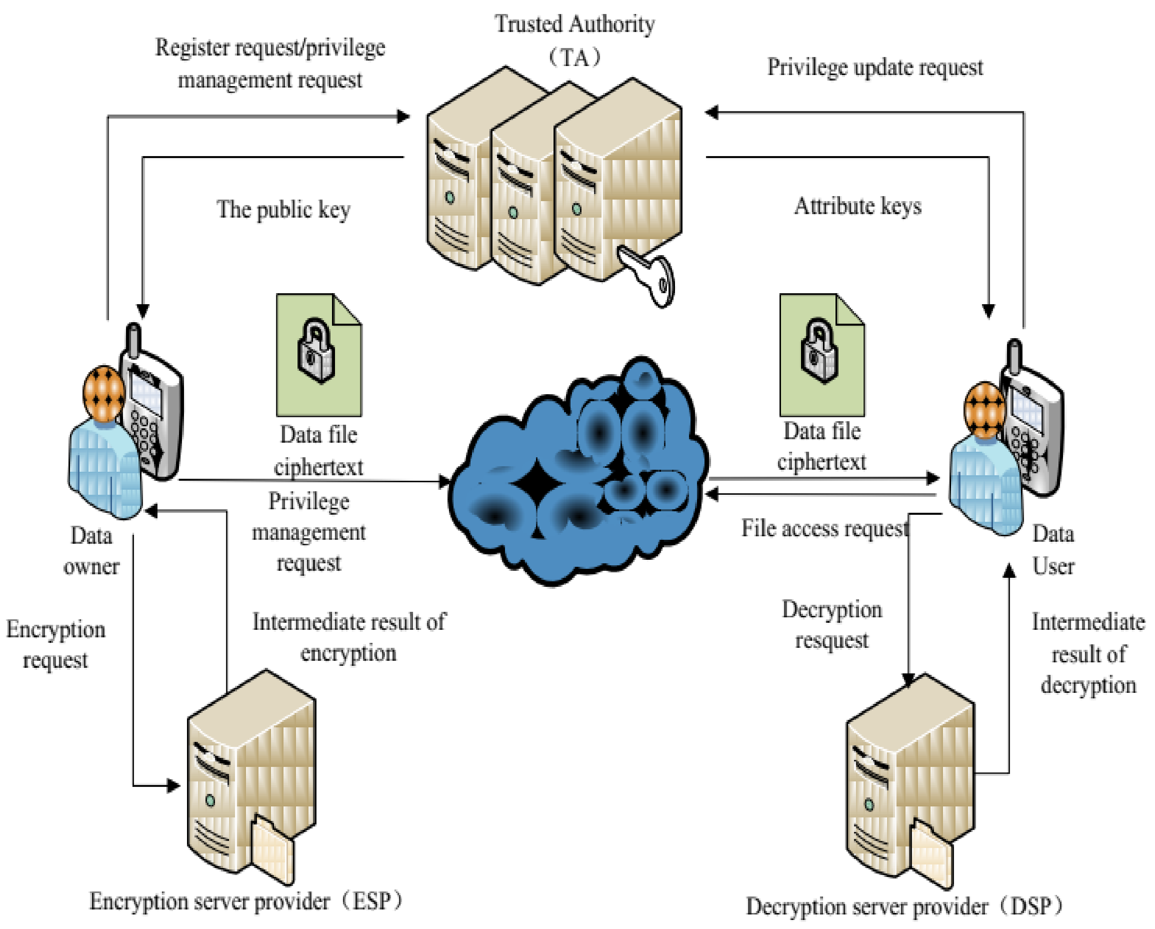

Trusted Authority (TA)
Register request/privilege management request
Privilege update request
The public key
Attribute keys
Data file ciphertext
Privilege management request
Data file ciphertext
File access request
Data owner
Data User
Encryption request
Intermediate result of encryption
Decryption resquest
Intermediate result of decryption
Encryption server provider（ESP）
Decryption server provider（DSP）

Figure 7.3: Architecture of LDSS

Modules

1. Data Owner (DO)
2. Data User (DU)
3. Trust Authority (TA)
4. Encryption Service Provider (ESP)
5. Decryption Service Provider (DSP)
6. Cloud Service Provider (CSP)

Module Description:

Data Owner (DO):

DO uploads data to the mobile cloud and share it with friends. DO determines the access control policies.

Data User (DU):

DU retrieves data from the mobile cloud.

Trust Authority (TA):

TA is responsible for generating and distributing attribute keys.

Encryption Service Provider (ESP):

ESP provides data encryption operations for DO.

Decryption Service Provider (DSP):

DSP provides data decryption operations for DU.

Cloud Service Provider (CSP):

CSP stores the data for DO. It faithfully executes the operations requested by DO, while it may peek over data that DO have stored in the cloud.

7.3.1 Advantages of Proposed system

1. Greatly reduce the computational overhead on client-side mobile devices.

The lazy re-encryption and description field of attributes to reduce the revocation overhead when dealing with the user revocation problem.

Chapter 8
Implementation

8.1 Cloud Server Files

Cloudserver.java:

```
package com;

import java. awt. BorderLayout;

import java.awt. Color;

import java.awt. Container;

import java.awt. Font;

import javax. swing. JFrame;

import javax. swing. Label;

import javax. swing. Label;

import javax. swing. UIManager;

import javax. swing. JTextArea;

import javax. swing. JScrollPane;

import java.net. Socket;

import java.net. ServerSocket;

import java.net. InetAddress;

import javax. swing. JOptionPane;

public class CloudServer extends JFrame implements Runnable {

        JLabel l1;
        Font f1, f2;

        JPanel p1, p2, p3;

        Thread thread;

        JTextArea area;

        JScrollPane jsp;

        ServerSocket server;

        RequestHandler rh;

        static double time;

    public void start () {

        try {

            server = new ServerSocket (1111);

            area. Append ("Cloud Service Provider Started\n\n");

            while(true) {

                Socket socket = server. Accept ();

                socket. setKeepAlive (true);

                InetAddress address=socket. getInetAddress ();

                String ipadd=address. toString ();

                area. Append ("Connected Computers: "+ipadd. Substring
                (1, ipadd. length ()) +"\n");

                rh = new RequestHandler (socket, area);

                rh. Start ();

            }
        } catch (Exception e) {

                e. printStackTrace ();

        }

}

public CloudServer () {

        setTitle ("Cloud Service Storage Provider");

        getContentPane (). setLayout (new BorderLayout ());

        f1 = new Font ("Monospaced”, Font.BOLD,22);

        p1 = new JPanel ();

        l1 = new JLabel("<HTML><BODY><CENTER>Cloud Service Provider
        Application</CENTER></BODY></HTML>". toUpperCase ());

        l1. setFont (this. f1);

        l1. setForeground (new Color (125,54,2));

        p1. setBackground (Color. black);

        p1.add(l1);

        f2 = new Font (“Courier New”, Font.PLAIN,16);

        p2 = new JPanel ();

        p2. setLayout (new BorderLayout ());

        area = new JTextArea ();

        area. setFont (f2);

        area. setEditable (false);

        jsp = new JScrollPane(area);
```

```
          p2.add (jsp, BorderLayout. CENTER);
          getContentPane (). add (p1, BorderLayout. NORTH);
          getContentPane (). add (p2, BorderLayout. CENTER);
          thread = new Thread(this);
          thread. Start ();
}
public void run () {
          try {
                    while(true) {
                                        l1. setForeground (Color. white);
                                        thread. sleep (500);
                                        l1. setForeground (Color. red);
                                        thread. sleep (500);
                              }
          } catch (Exception e) {
                    e. printStackTrace ();
          }
}
public static void main (String a []) throws Exception {
         UIManager. SetLookAndFeel (UIManager.getLookAndFeelClassName());
          CloudServer screen = new CloudServer ();
```

```
        screen. Pack ();

        screen. setExtendedState (JFrame. MAXIMIZED_BOTH);

        screen. setVisible (true);

        new Server Thread (screen);

}

}
```

DBCon.java:

```
package com;

import java. sql. Connection;

import java. sql. DriverManager;

import java. sql. PreparedStatement;

import java. sql. ResultSet;

import java. util. Calendar;

import java. sql.  Statement;

import java. util. ArrayList;

import java. io. FileWriter;

import java. io. ObjectInputStream;

import java. io. ObjectOutputStream;

import java. net. Socket;

import java. io. FileOutputStream;

import java. io. File;
import java. io. FileInputStream;

import java. util. List;

public class DBCon {

    private static Connection con;

    static String check = "none";

    public static Connection getCon () throws Exception {

          try {

                    Class.  forName ("com.  Mysql. jdbc. Driver");

                    Con=DriverManager. GetConnection("jdbc:mysql://LDSS,” root”);

               } catch (Exception e) {

                    e. printStackTrace ();

          }

          return con;

}

public static String register (String [] input) throws Exception {

          String msg="Error in registration";

          con = getCon ();

          Statement stmt= con.  createStatement ();

          ResultSet rs=stmt. executeQuery ("select username from users where
          username='"+input [ 0] +"'");

           If (rs. Next ()) {

                     msg = "Username already exist";
```

```
        }
    else {
            PreparedStatement stat= con. prepareStatement ("insert into users
            values(?)");
            stat. setString (1, input [0]);
            stat. setString (2, input [1]);
            stat. setString (3, input [2]);
            stat. setString (4, input [3]);
            stat. setString (5, input [4]);
            int i= stat. executeUpdate ();
            if (i > 0) {
                        msg = "success";
                  }
        }
      return msg;
    }
    public static String shareDetails (String owner, String csp, String tag, String
    file, String policy) throws Exception {
      String msg="Error in registration";
      String refrence = "none";
      check = "none";
      con = getCon ();
```

```
  Statement stmt= con. createStatement ();
  ResultSet rs = stmt. executeQuery ("select csp_name, owner_name, file_name,
  share_att from share where filetag='"+tag+"'");
   If (rs. Next ()) {
       refrence = rs. getString (1) +","+ rs. getString (2) +","+ rs. GetString (3);
       check = rs. getString (4);
     }
PreparedStatement stat= con. prepareStatement ("insert into share values(?)");
stat. setString (1, owner);
stat. setString (2, csp);
stat. setString (3, policy);
stat. setString (4, file);
stat. setString (5, tag);
stat. setString (6, refrence);
int i=stat. executeUpdate ();
if (i > 0)
                    msg = "success";
          return refrence;
}
public static String userLogin (String input []) throws Exception {
    String msg="invalid login";
    con = getCon ();
    Statement stmt=con. createStatement ();
    ResultSet rs=stmt. executeQuery ("select username from users where
    username='"+input [0] +"' and password='"+ input [1] +"'");
    If (rs. Next ()) {
            msg = rs. getString (1);
      }
         rs. Close ();
         stmt. Close ();
    return msg;
}
}
```

RequestHandler.java:

```
package com;
import java. io. ObjectInputStream;
import java. io. ObjectOutputStream;
import java. net. Socket;
import javax. swing. JTextArea;
import java. io. FileOutputStream;
import java. io. File;
import java. io. FileInputStream;
public class RequestHandler extends Thread {
    Socket;
    ObjectOutputStream out;
    ObjectInputStream in;
     JTextArea area;
     public RequestHandler (Socket soc, JTextArea area) {
         socket=soc;
          this. Area =area;
          try {
                    out = new ObjectOutputStream (socket. getOutputStream ());
                    in = new ObjectInputStream (socket. getInputStream ());
          } catch (Exception e) {
                    e. printStackTrace ();
          }
}
@Override
public void run () {
          try {
                    process ();
    } catch (Exception e) {
        e. printStackTrace ();
    }
}
```

```
public void process () throws Exception {
        Object input [] = (Object []) in. readObject ();
        String type=(String) input [0];
        If (type. equals ("register")){
                String user = (String) input [1];
                String pass = (String) input [2];
                String contact = (String) input [3];
                String address = (String) input [4];
                String email = (String) input [5];
                String arr [] = {user, pass, contact, address, email};
                String msg = DBCon. Register(arr);
                If (msg. equals("success")) {
                        File = new File("users/"+user);
                        file. Mkdir ();
                        Object res [] = {msg};
                        area. Append (user+" registered at cloud server\n");
                        out. writeObject (res);
                        out. Flush ();
                } else {
                        Object res [] = {msg};
                        area. append (msg+"\n");
                        out. writeObject (res);
                        out. Flush ();
                }
        }
        If (type. Equals ("login")) {
                String user = (String) input [1];
                String pass = (String) input [2];
                String arr [] = {user, pass};
                String msg = DBCon. userLogin (arr);
                If (! msg. equals ("invalid login")) {
                        Object res [] = {msg};
                        area. append (user+" login to "+msg+"\n");
                        out. writeObject(res);
                        out. flush ();
                } else {
                        Object res [] = {"invalid login"};
                        out. writeObject (res);
                        out. flush ();
                }
        }
        If (type. Equals ("upload")) {
```

```
                String user = (String)input [1];

                String file = (String)input [2];

                byte enc [] = (byte []) input [3];

                up. Join ();

                Object res [] = {"success", file+" saved at cloud server"};

                area. Append (file+" saved at cloud server\n");

                out. writeObject (res);

                out. Flush ();

        }

If (type. Equals ("download")) {

                String user = (String) input [1];

                String file = (String) input [2];

                FileInputStream fin = new FileInputStream("users/"+user+"/"+file);

                byte enc [] = new byte [fin. Available ()];

                fin. Read (enc,0, enc. length);

                fin. close ();

                Object res [] = {enc};

                area. Append (file+" data in the cloud is sent to user from
                owner:"+user+" \n");

                out. writeObject (res);

                out. Flush ();

        }
```

```
}
}
```

ServerThread.java:

```
package com;
public class ServerThread extends Thread {
        CloudServer server;
public ServerThread (CloudServer server) {
        this. Server =server;
        start ();
}
public void run () {
        server. Start ();
}
}
```

Upload.java:

```
package com;
import java.io. File;
import java.io. IOException;
import java. util. ArrayList;
import java. util. List;
import com. amazonaws. services. s3. model. Bucket;
import com. amazonaws. services. s3. AmazonS3;
import com. amazonaws. services. s3. AmazonS3Client;
import com. amazonaws. auth. BasicAWSCredentials;
public class Upload extends Thread {
        String user;
        String msg="fail";
        File file;
public String get Message () {
        return msg;
}
public Upload (String user, File file) {
        this. user = user;
        this. file = file;
        start ();
}
public void run () {
        try {
                upload ();
        } catch (Exception e) {
                e. printStackTrace ();
        }
}
public void upload () throws Exception {
        BasicAWSCredentials credentials = new
        BasicAWSCredentials("AKIAJW53TXEKCLXZWEFQ","dxpGxQ8LLO
        Ma6BBySLZFYZWeE7vczCF7wq3AUav6");
        AmazonS3 s3Client = new AmazonS3Client(credentials);
        If (! s3Client.doesBucketExist("ldss -"+user)){
                Bucket bucket = s3Client.createBucket("ldss-"+user,"us-west-2");
        }
        s3Client.putObject("ldss-"+user, file. getName (), file);
        msg = "success";
}
}
```

8.2 Trusted Authority Files

Encrypt.java:

```
package com;
import cpabe. Cpabe;
public class Encrypt
{
public static boolean encrypt (String input, String enc, String policy, String
public_key) {
        boolean flag = false;
        try {
                Cpabe att = new Cpabe ();
                att. Enc (public_key, policy, enc, enc);
                flag = true;
        } catch (Exception e) {
                e. printStackTrace ();
                flag = false;
        }
        return flag;
}
}
```

RequestHandler.java:

```
package com;
import java.io. ObjectInputStream;
import java.io. ObjectOutputStream;
import java.net. Socket;
import javax. swing. JTextArea;
import java.io. FileOutputStream;
import java.io. File;
import java.io. FileInputStream;
public class RequestHandler extends Thread {
    Socket socket;

    ObjectOutputStream out;

    ObjectInputStream in;

    JTextArea area;

    public RequestHandler (Socket soc, JTextArea area) {

          socket=soc;

          this. Area =area;

          try {

                    out = new ObjectOutputStream (socket. getOutputStream ());

                    in = new ObjectInputStream (socket. getInputStream ());

          } catch (Exception e) {

        e. printStackTrace ();

    }

}

@Override

public void run () {

          try {

                   process ();

    } catch (Exception e) {

        e. printStackTrace ();

    }
```

```
}
public void process () throws Exception {
        Object input [] = (Object []) in. readObject ();
        String type=(String) input [0];
        If (type. equals ("register")) {
                String user = (String) input [1];
                SecretKey. generate Key (user);
                Object res [] = {"Secret key generated for user: "+user};
                area. Append ("Secret key generated for user: "+user+"\n");
                out. writeObject (res);
                out. Flush ();
        }
        If (type. Equals ("getkey")) {
                String user = (String) input [1];
                byte pub [] = new byte [fin. Available ()];
                fin. Read (pub,0, pub. length);
                fin. close ();
                fin = new FileInputStream("keys/"+user+"/master.txt");
                byte mas [] = new byte [fin. Available ()];
                fin. Read (mas,0, mas. length);
                fin. Close ();
                Object res [] = {pub, mas};

                area. Append ("Keys sent to encryption server for user: "+user+"\n");

                out. writeObject (res);

                out. Flush ();

        }

}

}
```

SecretKey.java:

```
package com;

import cpabe. Cpabe;

import java. io. File;

public class SecretKey {

public static boolean generateKey (String user) {

        boolean flag = false;

        try {

                File file = new File("keys/"+user);

                If (! file. exists ()) {

                        file. Mkdir ();

                }

                Cpabe att = new Cpabe ();

                String public_key = "keys/"+user+"/public.txt";
                String master_key = "keys/"+user+"/master.txt";
                att. setup (public_key, master_key);
                flag = true;
        } catch (Exception e) {
                e. printStackTrace ();
                flag = false;
        }
        return flag;
}
public static boolean privateKey (String user, String public_key, String
master_key, String private_key, String attributes) {
        boolean flag = false;
        try {
                Cpabe att = new Cpabe ();
                att. Keygen (public_key, private_key, master_key, attributes);
                flag = true;
        } catch (Exception e) {
                e. printStackTrace ();
                flag = false;
        }
        return flag;
}
```

ServerThread.java:

```
package com;
public class ServerThread extends Thread {
        TrustedAuthority server;
public ServerThread (TrustedAuthority server) {
        this. Server =server;
        start ();
}
public void run () {
        server. start ();
}
}
```

TrustedAuthority.java:

```
package com;
import java.awt. BorderLayout;
import java.awt. Color;
import java.awt. Container;
import java.awt. Font;
import javax. swing. JFrame;
import javax. swing. JLabel;
import javax. swing. JLabel;
import javax. swing. UIManager;
import javax. swing. JTextArea;
import javax. swing. JScrollPane;
import java. net. Socket;
import java. net. ServerSocket;
import java.net. InetAddress;
import javax.  swing. JButton;
import java.awt. event. ActionListener;
import java.awt. event. ActionEvent;
import javax. swing. JOptionPane;
import java.io. File;
public class TrustedAuthority extends JFrame implements Runnable {
        JLabel l1;
        Font f1, f2;
        JPanel p1, p2;
        Thread thread;
        JTextArea area;
        JScrollPane jsp;
        ServerSocket server;
        RequestHandler rh;
        public void start () {
```

```
        try {
                server = new ServerSocket (2222);
                area. Append ("Trusted Authority Center Started\n\n");
                while(true) {
                Socket socket = server. Accept ();
                socket. setKeepAlive (true);
                InetAddress address=socket. getInetAddress ();
                String ipadd=address. toString ();
                area. Append ("Connected Computers:"+ipadd. Substring (1, ipadd.
                Length ()) +"\n");
                rh = new RequestHandler (socket, area);
                rh. start ();
                }
        } catch (Exception e) {
                e. printStackTrace ();
        }
}
public TrustedAuthority () {
        setTitle ("Trusted Authority Center");
        getContentPane (). setLayout (new BorderLayout ());
        f1 = new Font ("Monospaced", Font.BOLD,22);
        p1 = new JPanel ();
        l1 = new JLabel("<HTML><BODY><CENTER>Trusted Authority Key
        Generation Application</CENTER></BODY></HTML>". toUpperCase
        ());

        l1. setFont (this. f1);

        l1. setForeground (new Color (125,54,2));

        p1. setBackground (Color. black);

        p1.add(l1);

        f2 = new Font ("Courier New", Font.PLAIN,16);

        p2 = new JPanel ();

        p2. setLayout (new BorderLayout ());

        area = new JTextArea ();

        area. setFont (f2);

        area. setEditable (false);

        jsp = new JScrollPane(area);

        p2. add (jsp, BorderLayout. CENTER);

        getContentPane (). Add (p1, BorderLayout. NORTH);

        getContentPane (). Add (p2, BorderLayout. CENTER);

        thread = new Thread(this);

        thread. Start ();

}

public void run () {

        try {
```

```
                    while(true) {
                              l1. setForeground (Color. white);
                              thread. sleep (500);
                              l1. setForeground (Color. red);
                              thread. sleep (500);
                    }
          } catch (Exception e) {
                    e. printStackTrace ();
          }
}
public static void main (String a []) throws Exception {
UIManager.setLookAndFeel(UIManager.getLookAndFeelClassName());
          TrustedAuthority screen = new TrustedAuthority ();
          screen. Pack ();
          screen. setExtendedState (JFrame. MAXIMIZED_BOTH);
          screen. setVisible (true);
          new ServerThread(screen);
}
}
```

8.3 EncryptionDecryptionServer Files

EncryptionDecryptionServer.java:

```
package com;

import java.awt. BorderLayout;

import java.awt. Color;

import java.awt. Container;

import java.awt. Font;

import javax. swing. JFrame;

import javax. swing. JLabel;

import javax. swing. JLabel;

import javax. swing. UIManager;

import javax. swing. JTextArea;

import javax. swing. JScrollPane;

import java. net. Socket;

import java. net. ServerSocket;

import java. net. InetAddress;

import javax. swing. JButton;

import java.awt. event. ActionListener;

import java.awt. event. ActionEvent;

import javax. swing. JOptionPane;

import java.io. File;
```

```
public class EncryptionDecryptionServer extends JFrame implements Runnable
{
        JLabel l1;
        JButton b1;
        Font f1, f2;
        JPanel p1, p2, p3;
        Thread thread;
        JTextArea area;
        JScrollPane jsp;
        ServerSocket server;
        RequestHandler rh;
        static double time;
public void start () {
        try {
                server = new ServerSocket (3333);
                area. Append ("Encryption Decryption Server Application\n\n");
                while(true) {
                        Socket socket = server. Accept ();
                        socket. setKeepAlive (true);
                        InetAddress address= socket. getInetAddress ();
                        String ipadd=address. toString ();
```

```
                    area. Append ("Connected   Computers:"+ipadd. Substring
                    (1, ipadd. length())+"\n");

                    rh = new RequestHandler (socket, area);

                    rh. start ();

                }

            } catch (Exception e) {

                e. printStackTrace ();

            }

        }

        public EncryptionDecryptionServer () {

            setTitle ("Encryption Decryption Server Application");

            getContentPane (). setLayout (new BorderLayout ());

            f1 = new Font ("Monospaced", Font. BOLD, 22);

            p1 = new JPanel ();

            l1 = new JLabel("<HTML><BODY><CENTER>Encryption Decryption
            Server Application</CENTER></BODY></HTML>". toUpperCase ());

            l1. setFont (this. f1);

            l1. setForeground (new Color (125,54,2));

            p1. setBackground (Color. black);

            p1.add(l1);

            f2 = new Font ("Courier New", Font.PLAIN,16);

            p2 = new JPanel ();
            p2. setLayout (new BorderLayout ());
            area = new JTextArea ();
            area. setFont (f2);
            area. setEditable (false);
            jsp = new JScrollPane(area);
            p2. add (jsp, BorderLayout. CENTER);
            getContentPane (). Add (p1, BorderLayout. NORTH);
            getContentPane (). Add (p2, BorderLayout. CENTER);
            thread = new Thread(this);
            thread. Start ();
        }
        public void run () {
            try {
                while(true) {
                    l1. setForeground (Color. white);
                    thread. sleep (500);
                    l1. setForeground (Color. red);
                    thread. sleep (500);
                }
            } catch (Exception e) {
                e. printStackTrace ();
```

```
          }
}
EncryptionDecryptionServer screen = new EncryptionDecryptionServer ();
           screen. Pack ();
           screen. setExtendedState (JFrame. MAXIMIZED_BOTH);
           screen. setVisible (true);
           new ServerThread(screen);
}
}
```

Encrypt.java:

```
package com;
import cpabe. Cpabe;
public class Encrypt
{
public static boolean encrypt (String input, String enc, String policy, String public_key) {
          boolean flag = false;
          try {
                    Cpabe att = new Cpabe ();
                    att. Enc (public_key, policy, enc, enc);
                    flag = true;
          } catch (Exception e) {
                e. printStackTrace ();
                flag = false;
        }
        return flag;
}
}
```

Decrypt.java:

```
package com;
import cpabe. Cpabe;
import java. io. File;
import java. io. FileInputStream;
public class Decrypt {
        static String msg;
public static String getMsg () {
        return msg;
}
public static byte [] decrypt (String public_key, String private_key, String enc,
String dec_path) {
        byte b [] = null;
        try {
                File dec = new File(dec_path);
                Cpabe test = new Cpabe ();
                test. Dec (public_key, private_key, enc, dec. getPath ());
                FileInputStream fin = new FileInputStream(dec);
                b = new byte [fin. available ()];
                fin. Read (b, 0, b. length);
                fin. Close ();
                dec. delete ();
                msg = "success";
        } catch (Exception e) {
                e. printStackTrace ();
                b = "error". getBytes ();
                msg = "error";
        }
        return b;
}
}
```

GenerateKey.java:

```
package com;
import bswabe. Bswabe;
import bswabe. BswabeCph;
import bswabe. BswabeCphKey;
import bswabe. BswabeElementBoolean;
import bswabe. BswabeMsk;

import bswabe. BswabePrv;

import bswabe. BswabePub;

import bswabe. BswabeCph;

public class GenerateKey {

        static String [] attr = {"owner","consumer"};

        static BswabePub public_key;

public static BswabePrv generateKey () throws Exception {

        public_key = new B

        swabePub ();

        BswabeMsk master_key = new BswabeMsk();

        Bswabe. Setup (public_key, master_key);

        BswabePrv private_key = Bswabe. Keygen (public_key, master_key, attr);

        return private_key;

}

public static BswabeCph encrypt (String policy, BswabePub pu) throws
Exception {

        public_key = pu;

        BswabeCphKey crypted = Bswabe. Enc (public_key, policy);

        BswabeCph cph = crypted. cph;

        return cph;

}
public static boolean decrypt (BswabePub pub, BswabePrv prv, BswabeCph cph)
throws Exception {

        boolean flag=false;

        BswabeElementBoolean result = Bswabe. Dec (pub, prv, cph);

        If (result. b == true)

                flag = true;

        return flag;

}

}
```

Secretkey.java:

```
package com;

import cpabe. Cpabe;

import java. io. File;

public class SecretKey {

public static boolean generateKey (String user) {

        boolean flag = false;

        try {

                File file = new File("keys/"+user);

                If (! file. Exists ()) {

                        file. Mkdir ();

                }

                Cpabe att = new Cpabe ();
                String public_key = "keys/"+user+"/public.txt";

                String master_key = "keys/"+user+"/master.txt";

                att. Setup (public_key, master_key);

                flag = true;

        } catch (Exception e) {

                e. printStackTrace ();

                flag = false;

        }

        return flag;

}

public static boolean privateKey (String user, String public_key, String
master_key, String private_key, String attributes) {

        boolean flag = false;

        try {

                Cpabe att = new Cpabe ();

                att. Keygen (public_key, private_key, master_key, attributes);

                flag = true;

        } catch (Exception e) {

                e. printStackTrace ();

                flag = false;

        }

        return flag;
```

```
}
}
```

ServerThread.java:

```
package com;
public class ServerThread extends Thread {
        EncryptionDecryptionServer server;
public ServerThread (EncryptionDecryptionServer server) {
        this. Server =server;
        start ();
}
public void run () {
        server. Start ();
}
}
```

JSP Files:

Upload. jsp:

```
<%@page import="com. DBCon"%>
<%@page import="java. Sql. Connection"%>
<%@page import="java. Sql. ResultSet"%>
<%@page import="java. Sql. Statement"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link href="style.css" rel="stylesheet" type="text/css" />
<script language="javascript">
        function validate(formObj)
        {
        If (formObj.t1. value. length==0)
        {
        Alert ("Please upload file");
        formObj.t1. focus ();
        return false;
        }
        formObj. actionUpdateData. value="update";
        return true;
        }
        </script>
</head>
<body>
<div class="main">
```

```
<div class="main_resize">

<div class="header">

<div class="logo">

<h1 style="font-size:30Px; text-align: center;"><span>A Lightweight Secure
Data Sharing Scheme for Mobile Cloud Computing</span></h1>
</div>

</div>

<div class="content">

<div class="content_bg">

<div class="menu_nav">

<ul>

        <li><a href="UserScreen.jsp">Home</a></li>

        <li class="active"><a href="Upload. Jsp">Upload File</a></li>

        <li><a href="ViewShare.jsp">Download File</a></li>

        <li><a href="UserRevocation.jsp">Revoke users</a></li>

         <li><a href="Graph. Jsp">Overhead Graph</a></li>

         <li><a href="Logout. Jsp">Logout</a></li>

    </ul>

    </div>

    <h2 style=" text-align: center; color: red;"><b>Upload File</b></h2>

    <table class="hbg" width="100%" height="100%"> <tr>

    <td><div style="text-align: center;"><img src="images.jpg" alt="" /></div>
   </td>
```

```
<td>

<center>

<form name="f1" method="post" action="Upload" enctype="multipart/form-
data" onsubmit="return validate(this);"><br/>

<%

        String res = request. getParameter ("t1");

        If (res! = null) {

                        out. Println ("<center><font face= verdana
                        color=red>"+res+"</center></font>");

        }%>

<table align="center" width="40" >

<tr><td align ="left"><font size="2">Share& nbsp; With:</font></td>

<td>&nbsp;<select name="t2" multiple>

<%

        String owner = session. getAttribute ("user"). toString ();

        Connection con = DBCon. getCon ();

        Statement stmt = con. createStatement ();

        ResultSet rs = stmt. executeQuery ("select username from users where
        username! = '"+owner+"'");

        While (rs. Next ()) {

                String user = rs. getString (1);

                %>

<option value="<%=rs. getString (1) %>"><%=rs. getString (1) %></option>
<%}%>
</select>
</td>
</tr>
<tr><td></td><td><input type="submit" value="Upload"></td>
</table>
</td> </tr> </table>
</div>
</div>
</body>
</html>
```

Download. jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8”
pageEncoding="UTF-8"%>
<%@page import="java.net. Socket"%>
<%@page import="com. DBCon"%>
<%@page import="java.io. ObjectInputStream"%>
<%@page import="java.io. ObjectOutputStream"%>
<%@page import="java.io. FileInputStream"%>
<%@page import="java.io. File"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>PHR</title>
</head>
<body>
<%!
byte b [];
%>
<%
String datauser = session. getAttribute ("user"). toString (). Trim ();
String user = request. getParameter ("t1");
String file = request. getParameter("t2");
String share = request. getParameter ("t3");
System. out. Println (datauser+" "+user+" "+ share. indexOf (datauser));
If (share. indexOf(datauser)! = -1) {
Socket socket = new Socket("localhost",3333);
ObjectOutputStream oout = new ObjectOutputStream (socket. getOutputStream
());
Object req [] = {"decrypt", user, file};
oout. writeObject (req);
oout. Flush ();
ObjectInputStream oin = new ObjectInputStream (socket. getInputStream ());

Object res [] = (Object []) oin. readObject ();

String msg = (String) res [0];

byte dec [] = (byte []) res [1];

if (msg. equals("success")) {

        response. setHeader ("Content-Disposition", "attachment; filename=\""
        +file + "\"");

        response. setHeader ("Content-Type", "application/octet-stream;");

        java. io. Output Stream os=response. getOutputStream ();

        os. Write (dec,0, dec. length);

        os. flush ();

        os. close ();

} else {

        out. Println ("U dont have access control");

}

} else {

        out. Println ("U dont have access control!");

}

%>

</body>

</html>
```

Login. jsp:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title></title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="style.css" rel="stylesheet" type="text/css" />

<script language="javascript">

        function validate(formObj)

        {

        If (formObj.t1.value. length ==0)

        {

        Alert ("Please Enter User ID");

        formObj. t1. Focus ();

        return false;

        }

        If (formObj.t2. value. Length==0)

        {

        Alert ("Please Enter Password");

        formObj.t2. focus ();

        return false;
```

```
          }
          formObj. actionUpdateData. value="update";
          return true;
          }
          </script>
</head>
<body>
<div class="main">
<div class="main_resize">
<div class="header">
<div class="logo">
<h1 style="font-size:30Px; text-align: center;">
<span>A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing
</span><small></small></h1>
</div>
</div>
<div class="content">
<div class="content_bg">
<div class="menu_nav">
          <ul>
            <li><a href="index. jsp">Home</a></li>
            <li class="active"><a href="Login. jsp">Login</a></li>
            <li><a href="Register. jsp">Register</a></li>
```

```
        </ul>

      </div>

    <div class="hbg" style="text-align: center;"><img src="images/cropped.jpg" alt="" /></div>

    <center>

    <form name="f1" method="post" action="Login" onsubmit="return”

    validate(this);">

    <h2 style="font-size:18Px; text-align: center; color: red;"><b>User Login Screen</b></h2>

        <%

        String res = request. getParameter ("t1");

        If (res! = null) {

              out. Println ("<font face=verdana color=red>"+res+</font>");
        }%>

   <table align="center" width="40" >

   <tr><td><b>Username</b></td><td><input type="text" name="t1" style="font-family: Comic Sans MS" size=30/></td></tr>

   <tr><td><b>Password</b></td><td><input type="password" name="t2" style="font-family: Comic Sans MS" size=30/></td></tr>

    <tr><td></td><td><input type="submit" value="Login"></td>

   </table>

   </div>

   </div>
```

```
</body>
</html>
```

Logout. jsp:

```
<%
session. Invalidate ();
%>
<jsp: forward page="index. jsp" />
```

Index. jsp:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div class="main">
  <div class="main_resize">
    <div class="header">
      <div class="logo">
      <h1 style="font-size:30Px; text-align: center;"><span>A Lightweight
      Secure Data Sharing Scheme for Mobile Cloud Computing
      </span><small></small></h1>
      </div>

    </div>

    <div class="content">

      <div class="content_bg">

        <div class="menu_nav">

          <ul>

            <li class="active"><a href="index. jsp">Home</a></li>

            <li><a href="Login. jsp">Login</a></li>

            <li><a href="Register. jsp">Register</a></li>

          </ul>

        </div>

          <div class="hbg" style="text-align: center;"><img src="images.png"
          alt="" /></div>
          <h2 style="font-size:18Px; text-align: center; color:
          red;"><span>ABSTRACT</span></h2>
       <p    align="justify"><font    size="2.5"    face="verdana"    color="blue">
          With the popularity of cloud computing, mobile devices can store/retrieve
personal data from anywhere at any time. Consequently, the data security
problem in mobile cloud becomes more and more severe and prevents further
development of mobile cloud. There are substantial studies that have been
conducted to improve the cloud security. However, most of them are not
applicable for mobile cloud since mobile devices only have limited computing
resources and power. Solutions with low computational overhead are in great
need for mobile cloud applications. </p>
```

```
<p align="justify"><font size="2.5" face="verdana" color="blue">    Proposed system, A Lightweight Data Sharing Scheme (LDSS) For Mobile Cloud Computing adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. LDSS moves a large portion of the computationally intensive access control tree transformation in CP-ABE from mobile devices to external proxy servers. Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program-based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments. </p>

</body>

</html>
```

Revocation. jsp:

```
<%@page import="java. sql. Connection "%>

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

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

<%@page import="com. DBCon "%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title></title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="style.css" rel="stylesheet" type="text/css" />
```

```
</head>
<body>
<div class="main">
 <div class="main_resize">
  <div class="header">
   <div class="logo">
     <h1 style="font-size:30Px; text-align: center;"><span>A Lightweight
     Secure Data Sharing Scheme for Mobile Cloud Computing </span></h1>
   </div>
  </div>
  <div class="content">
   <div class="content_bg">
    <div class="menu_nav">
     <ul>
      <li><a href="UserScreen.jsp">Home</a></li>
      <li><a href="Upload. jsp">Upload File</a></li>
      <li><a href="ViewShare.jsp">Download File</a></li>
      <li class="active"><a href="Revocation. jsp">Revoke users</a></li>
      <li><a href="Graph. jsp">Overhead Graph</a></li>
      <li><a href="Logout. jsp">Logout</a></li>
     </ul>
    </div>
```

```
<br/>

<center><h2><font color="red";><b><center>Revocation
Screen<center></b></font></h2>

<h3><font size="5" color="black";><%if (request. getParameter("t1")! = null)
%>

<%=request. getParameter ("t1") %>!

</center>

<%

        String res = request. getParameter ("t1");

        If (res! = null) {

        out.println("&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;");

<table align="center" width="40" >

        <tr><td><b>File& nbsp; Name</b></td><td><select name="t1">

        <%

        String owner = session. getAttribute("user"). toString ();

        Connection con = DBCon. getCon ();

        Statement stmt = con. createStatement ();

        ResultSet rs = stmt. executeQuery ("select file_name from share where
owner_name='"+owner+"'");

        While (rs. Next ()) {%>

        <%}%>

        </select>
```

```
 </td></tr>
</tr>
</table>
</body>
</html>
</body>
</html>
```

UserRevocation.jsp:

```
<%@page import="java. sql. Connection"%>
<%@page import="java. sql. ResultSet"%>
<%@page import="java. sql. Statement"%>
<%@page import="com.  DBCon"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div class="main">
```

```
 <div class="main_resize">

  <div class="header">

   <div class="logo">

     <h1 style="font-size:30Px; text-align: center;"><span>A Lightweight
     Secure Data Sharing Scheme for Mobile Cloud Computing </span></h1>
  </div>
  </div>

  <div class="content">

   <div class="content_bg">

    <div class="menu_nav">

     <ul>

     <li><a href="UserScreen.jsp">Home</a></li>

      <li><a href="Upload. Jsp">Upload File</a></li>

      <li><a href="ViewShare.jsp">Download File</a></li>

      <li><a href="Graph. Jsp">Overhead Graph</a></li>

      <li><a href="Logout. Jsp">Logout</a></li>

     </ul>

    </div>

     <br/>

     <center>

     <h3><font size="5" color="black";><%if (request. getParameter ("t1")! =
null) %>

     <%= session. getAttribute ("user") %>! </font></h3>
```

```
        </center>

        <%

        String res = request. getParameter ("t1");

        If (res! = null) {

        out.println("&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
        nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
        nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;");

        }%>

<form    name="f1"    method="post"    action="UserRevocation.jsp"    on
submit="return validate(this);"><br/>

<table align="center" width="40" >

        String owner = session. getAttribute ("user"). ToString ();

        Connection con = DBCon. GetCon ();

        Statement stmt = con. createStatement ();

        ResultSet rs = stmt. executeQuery ("select file_name from share where
        owner_name='"+owner+"'");

</select>

</td></tr>

<tr><td></td><td><input type="submit" value="Submit"></td></tr>

</table>

</body>

</html>

</body>
```

```
</html>
```

UserScreen.jsp:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title></title>

<meta http-equiv="content-type" content="text/html; charset=utf-8" />

<link href="style.css" rel="stylesheet" type="text/css" />

</head>

<body>

<div class="main">

  <div class="main_resize">

    <div class="header">

      <div class="logo">

          <h1 style="font-size:30Px; text-align: center;"><span>A Lightweight
          Secure Data Sharing Scheme for Mobile Cloud Computing
          </span><small></small></h1>
      </div>

    </div>

    <div class="content">

      <div class="content_bg">

        <div class="menu_nav">
```

```
          <ul>
            <li class="active"><a href="UserScreen.jsp">Home</a></li>
           <li><a href="Upload. Jsp">Upload File</a></li>
           <li><a href="ViewShare.jsp">Download File</a></li>
           <li><a href="UserRevocation.jsp">Revocation</a></li>
           <li><a href="Graph. Jsp">Overhead Graph</a></li>
           <li><a href="Logout. Jsp">Logout</a></li>
          </ul>
        </div>
        <br/>
        <center>
        <h2 style="font-size:30Px; color: red;"><span>Welcome <%if (request.
        GetParameter("t1")! = null) %>
          <%=session. GetAttribute("user") %>! </span></h2>
          </center>
          <br/>
   </body>
</html>
```

Graph. Jsp:

```
<%@page import="org. jfree.ui. RefineryUtilities"%>
<%@page import="com. Chart"%>
<%
```

```
Chart chart1 = new Chart ("Overhead Chart");
chart1.pack();
RefineryUtilities.centerFrameOnScreen(chart1);
chart1.setVisible(true);
response. SendRedirect("UserScreen.jsp");
 %>
```

ViewShare.jsp:

```
<%@page import="java. sql. Connection"%>
<%@page import="java. sql. ResultSet"%>
<%@page import="java. sql. Statement"%>
<%@page import="com.  DBCon"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div class="main">
  <div class="main_resize">
```

```
<div class="header">

  <div class="logo">

      <h1 style="font-size:30Px;text-align:center;"><span>A Lightweight
      Secure Data Sharing Scheme for Mobile Cloud Computing
      </span><small></small></h1>
  </div>

</div>

<div class="content">

  <div class="content_bg">

    <div class="menu_nav">

      <ul>

        <li><a href="UserScreen.jsp">Home</a></li>

        <li><a href="Upload. Jsp">Upload File</a></li>

        <li class="active"><a href="ViewShare.jsp">Download File</a></li>

        <li><a href="UserRevocation.jsp">Revocation</a></li>

        <li><a href="Graph. Jsp">Overhead Graph</a></li>

        <li><a href="Logout. Jsp">Logout</a></li>

      </ul>

    </div>

    <div class="hbg" style="text-align: center;"><img src="images/down.jpg"
    alt="" />

    <br/>

    <center>
```

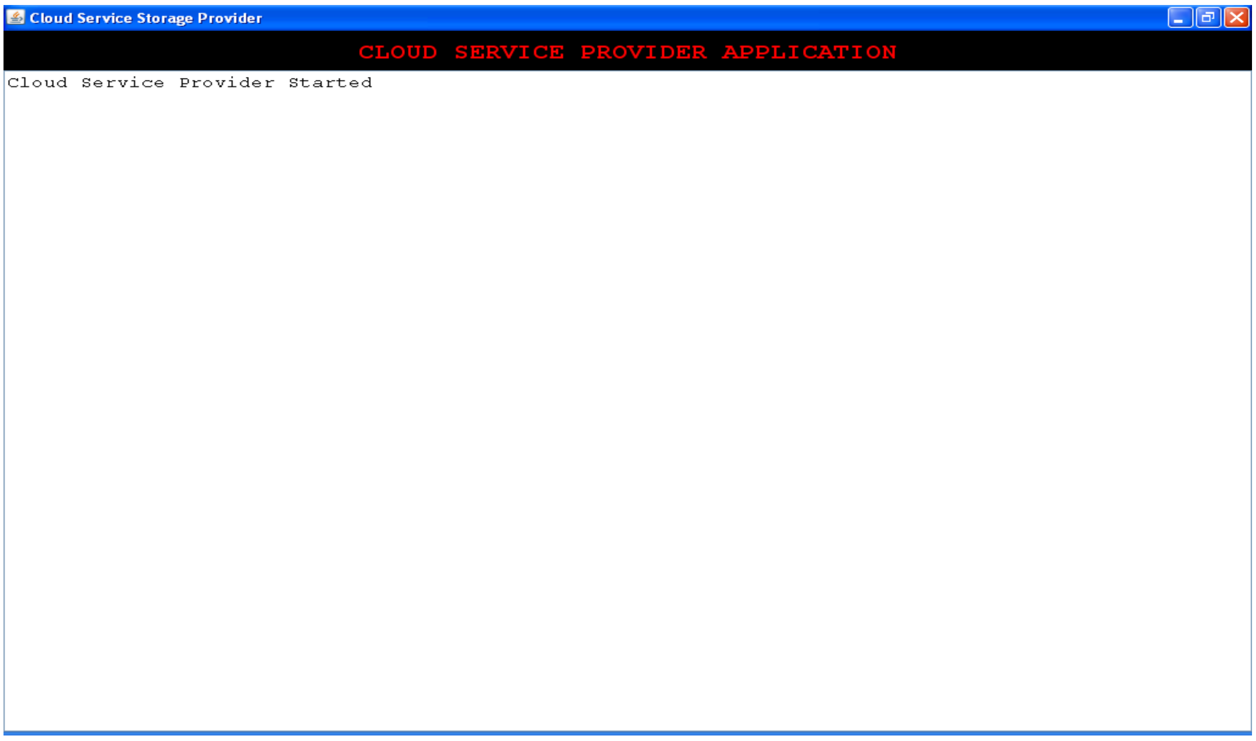
```
        <h2 style="font-size:20 Px;"><%if (request. getParameter("t1")!= null)%>

        <%=session. getAttribute ("user”) % >! </h2>

        </center>

        <%

        String res = request. getParameter ("t1");

        If (res! = null) {

        out.println("&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
        nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
        nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;");

<th>Download</th>

<tr>

        <%

        String owner = session. getAttribute ("user"). toString ();

        Connection con = DBCon. getCon ();

        Statement stmt = con. createStatement ();

        ResultSet rs = stmt. executeQuery ("select * from share");

        While (rs. Next ()) {

                String user = rs. getString (1);

                String att = rs. getString (2);

                String policy = rs. getString (3);

                String file = rs. getString (4);

                %>
        <tr><td><font size="3" color="black"><%=user%></td>

        <td><font size="3" color="black"><%=att%></td>

        <td><font size="3" color="black"><%=policy%></td>

        <td><font size="3" color="black"><%=file%></td>

        <td><a
href="Download.jsp?t1=<%=user%>&t2=<%=file%>&t3=<%=policy%>"><fo
nt size="3" color="black">Click Here</font></a></td>

        <%}%>

        </tr>

        </table>

        </div>

        </body>

</html>

</body>

</html>
```

Chapter 9

Screenshots of Project

9.1 Screenshots of Project

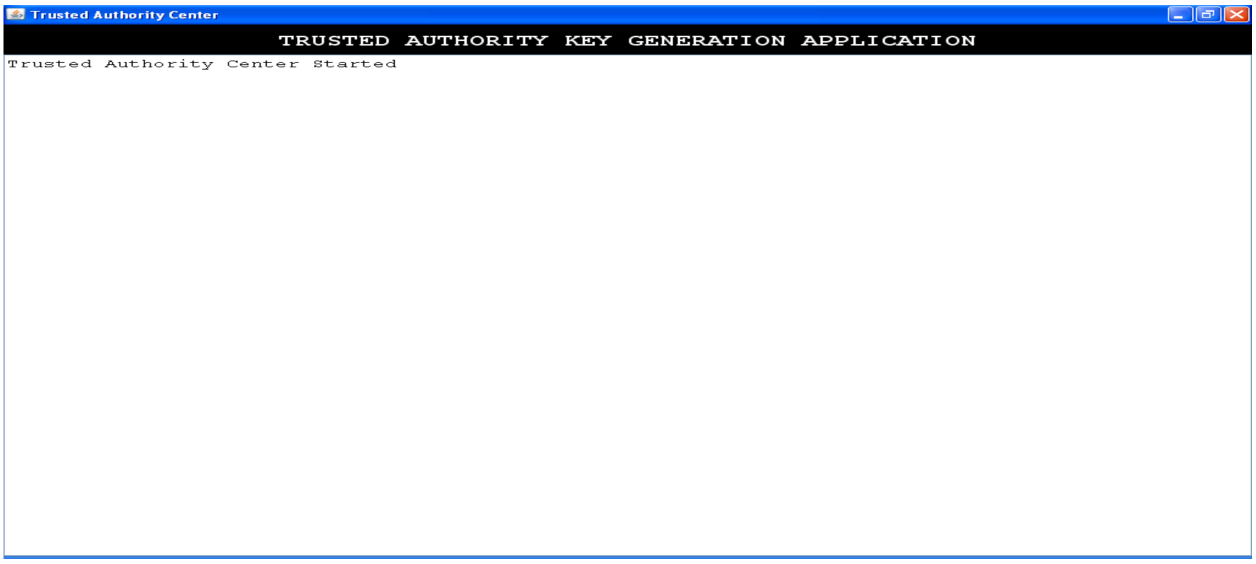
Double click on 'run.bat' file from LDSS/WEB-INF/CloudServer folder to start cloud server below screen


Cloud Service Storage Provider
CLOUD SERVICE PROVIDER APPLICATION

```
Cloud Service Provider Started
```

Figure 9.1.1: Screenshot of cloud server storage provider

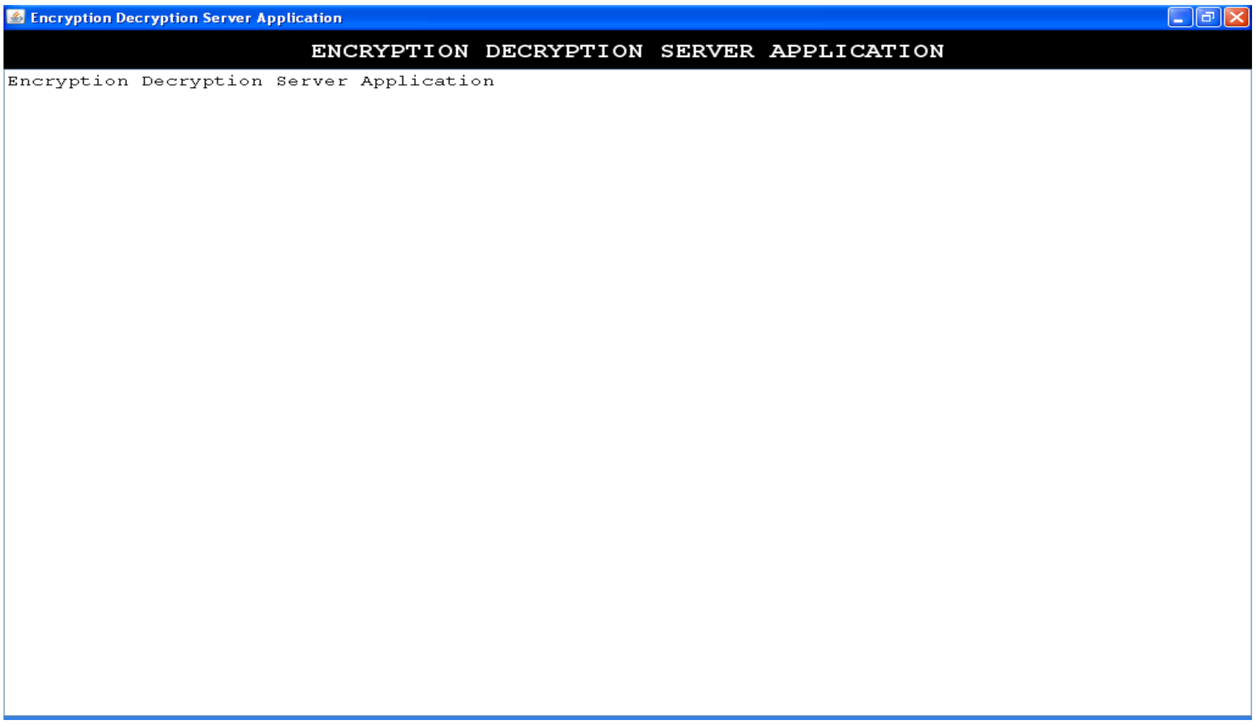
Double click on 'run.bat' file from LDSS/WEB-INF/TA folder to start Trusted Authority



```
TRUSTED AUTHORITY KEY GENERATION APPLICATION
Trusted Authority Center Started
```

Figure 9.1.2: Screenshot of Trusted Authority key generation application

Double click on 'run.bat' file from LDSS/WEB-INF/ Enc_Dec_Server folder to start encryption and decryption server



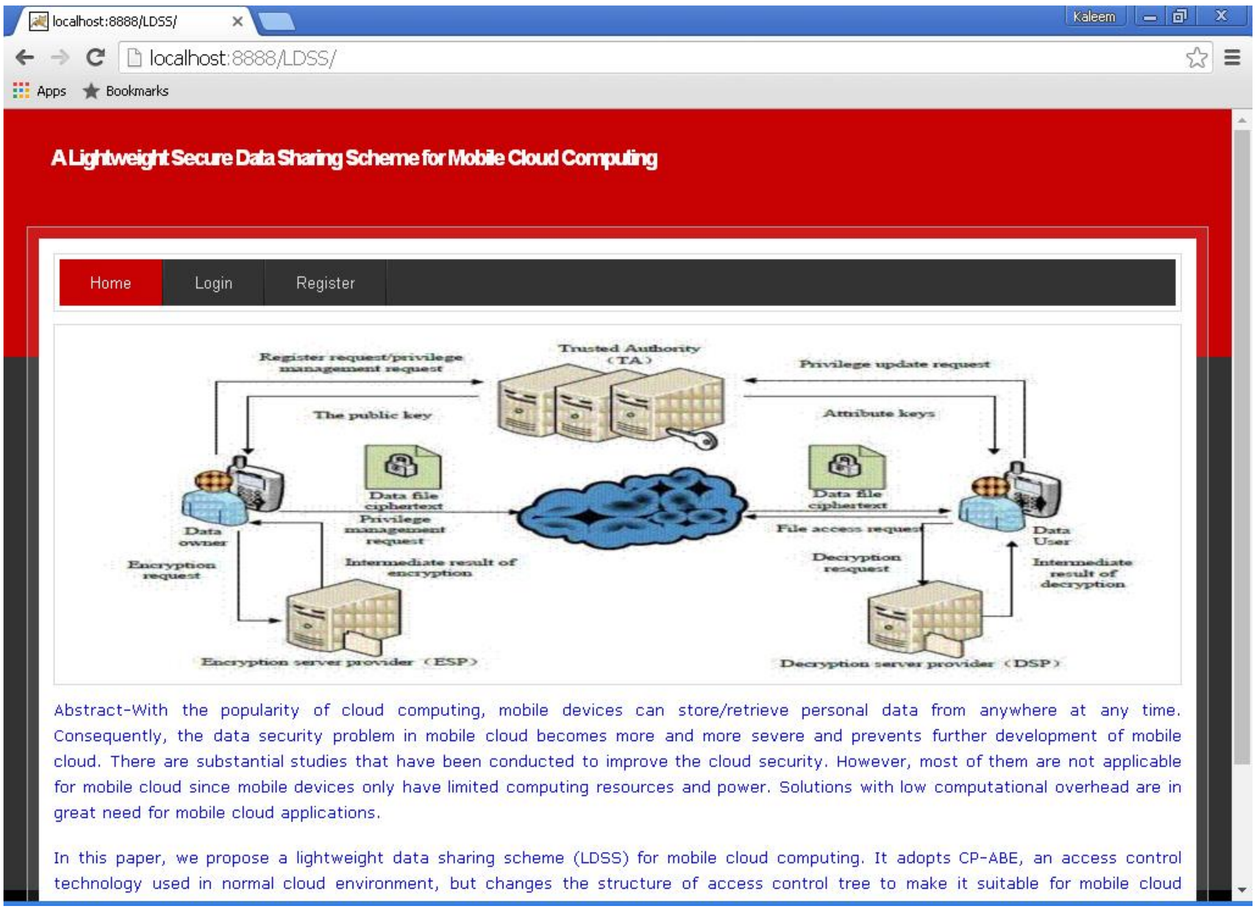
```
ENCRYPTION DECRYPTION SERVER APPLICATION
Encryption Decryption Server Application
```

Figure 9.1.3: Screenshot of encryption decryption server running

Now start tomcat server and open browser and enter URL as below

http://localhost:tomcat_server_port/LDSS

will get below screen



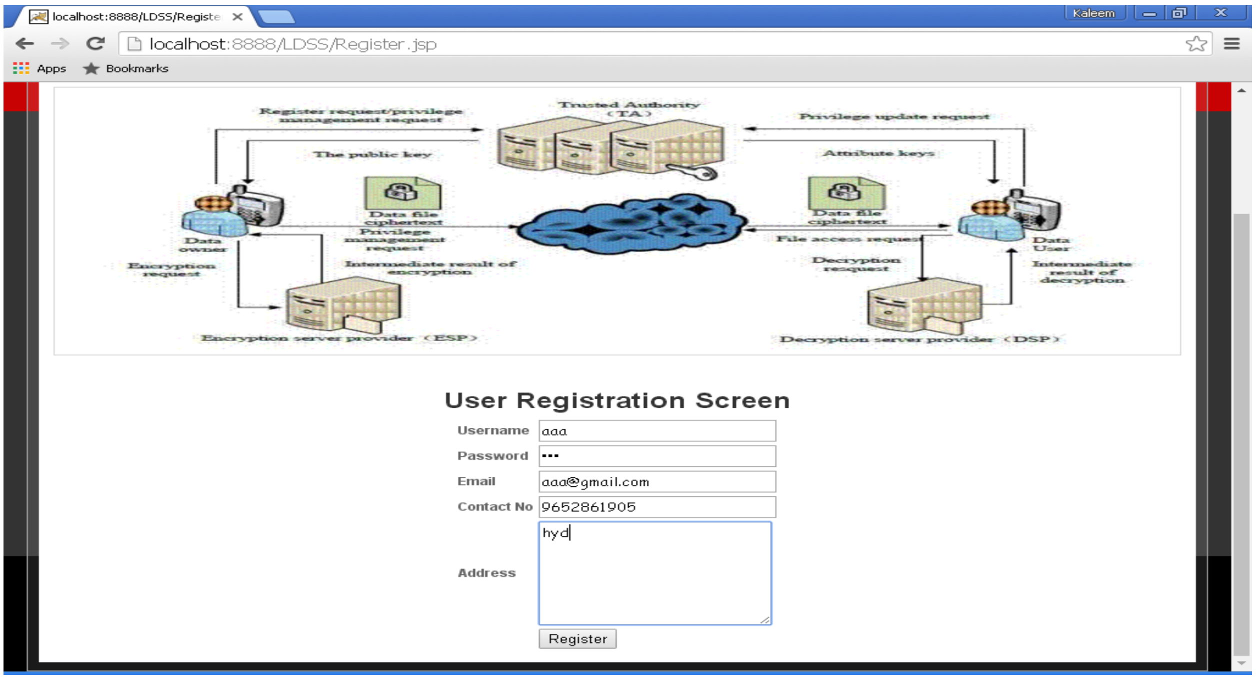
A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing

Abstract-With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in great need for mobile cloud applications.

In this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing. It adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud

Figure 9.1.4: Screenshot of running the application on local host

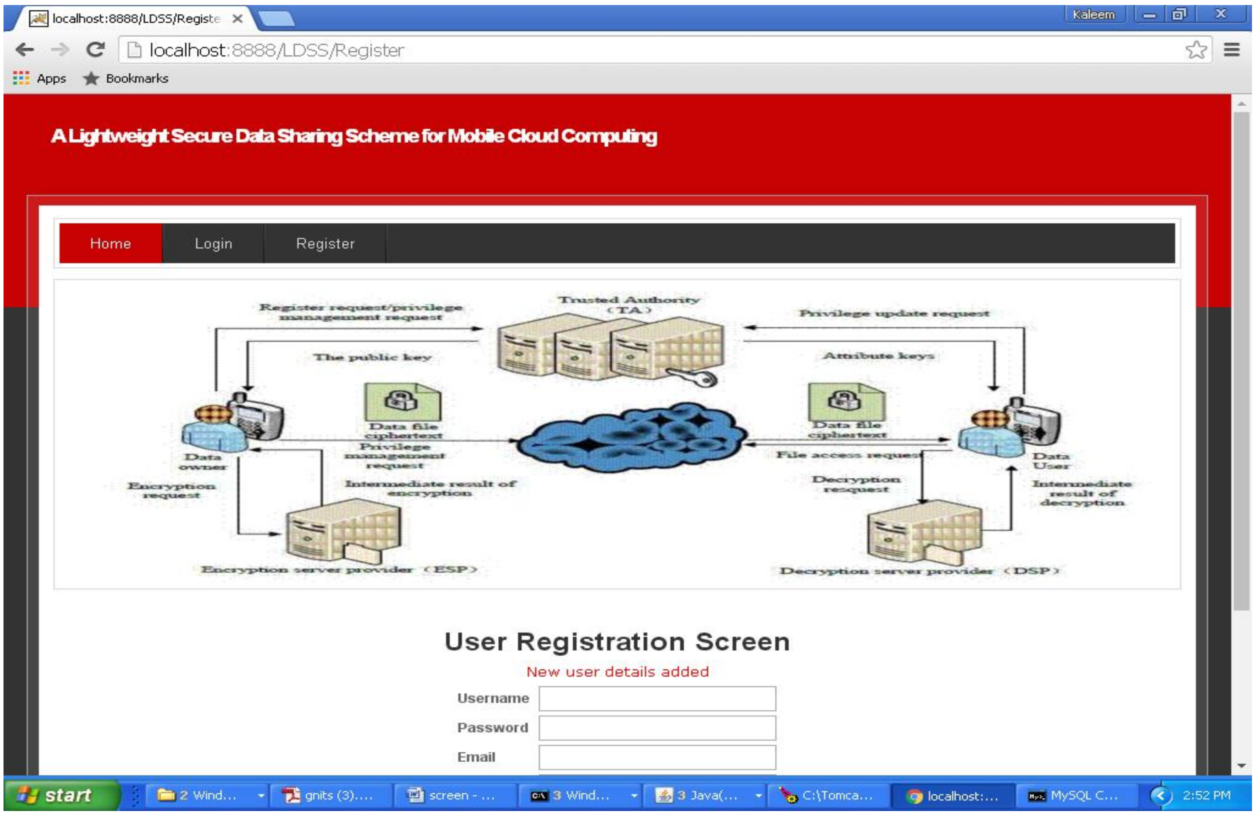
Now click on Register link and add some users


localhost:8888/LDSS/Register.jsp
Trusted Authority (TA)
Register request/privilege management request
The public key
Privilege update request
Attribute keys
Data owner
Data file ciphertext
Privilege management request
Encryption request
Intermediate result of encryption
File access request
Decryption request
Data User
Intermediate result of decryption
Encryption server provider (ESP)
Decryption server provider (DSP)
Register

User Registration Screen

Username	aaa
Password	•••
Email	aaa@gmail.com
Contact No	9652861905
Address	hyd

Figure 9.1.5: Screenshot of adding users


localhost:8888/LDSS/Register
A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing
Home
Login
Register
Trusted Authority (TA)
Register request/privilege management request
The public key
Privilege update request
Attribute keys
Data owner
Data file ciphertext
Privilege management request
Encryption request
Intermediate result of encryption
File access request
Decryption request
Data User
Intermediate result of decryption
Encryption server provider (ESP)
Decryption server provider (DSP)
start
2:52 PM

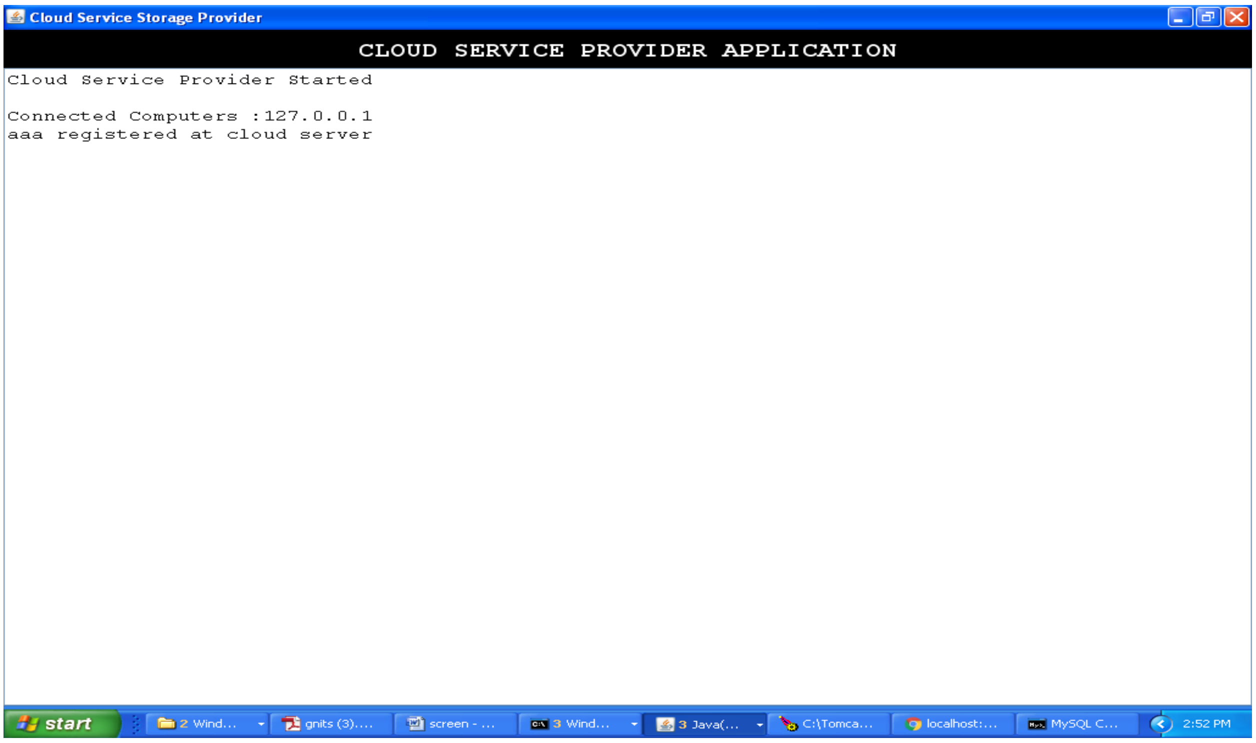
User Registration Screen

New user details added

Username	
Password	
Email	

Figure 9.1.6: Screenshot after adding the users

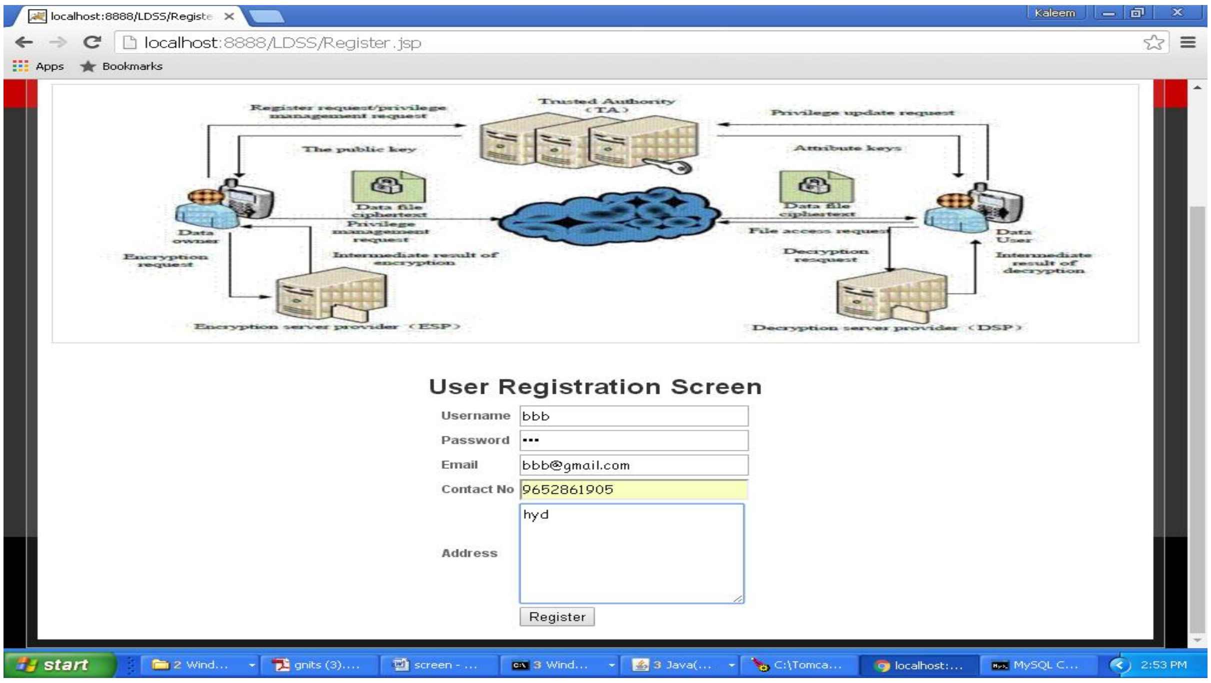
Will get user information at cloud server also



Cloud Service Storage Provider
CLOUD SERVICE PROVIDER APPLICATION
start
2:52 PM

```
Cloud Service Provider Started

Connected Computers :127.0.0.1
aaa registered at cloud server
```

Figure 9.1.7: Cloud service provider after adding users

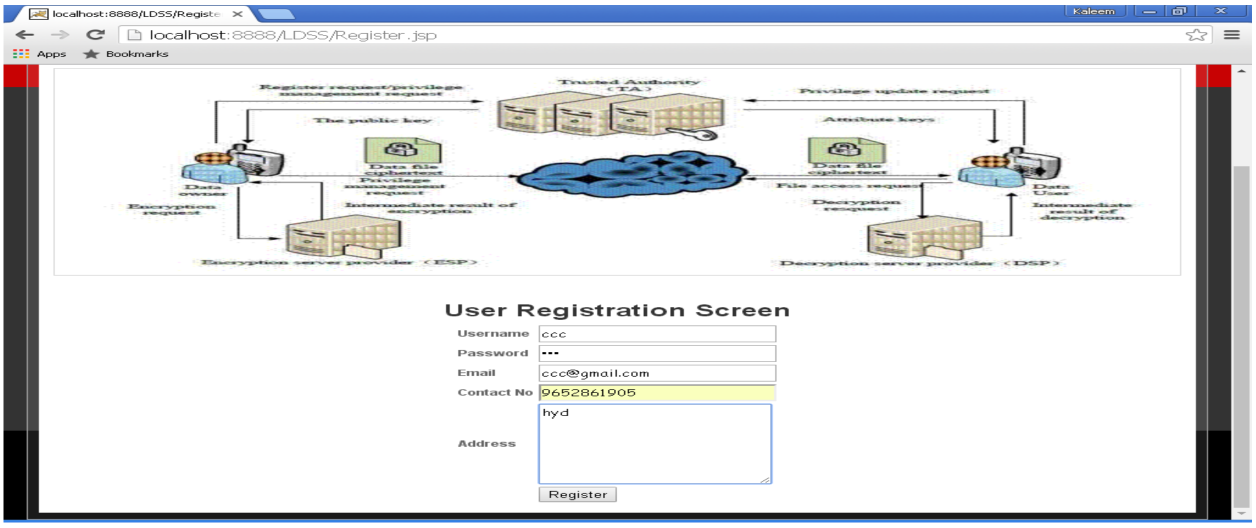


localhost:8888/LDSS/Register.jsp
Apps
Bookmarks
Register request/privilege management request
Trusted Authority (TA)
Privilege update request
The public key
Attribute keys
Data file ciphertext
Data file ciphertext
Privilege management request
File access request
Data owner
Data User
Encryption request
Intermediate result of encryption
Decryption request
Intermediate result of decryption
Encryption server provider (ESP)
Decryption server provider (DSP)
Register
start
2:53 PM

User Registration Screen

Username	bbb
Password	•••
Email	bbb@gmail.com
Contact No	9652861905
Address	hyd

Figure 9.1.8: Screenshot of User registration


localhost:8888/LDSS/Register.jsp

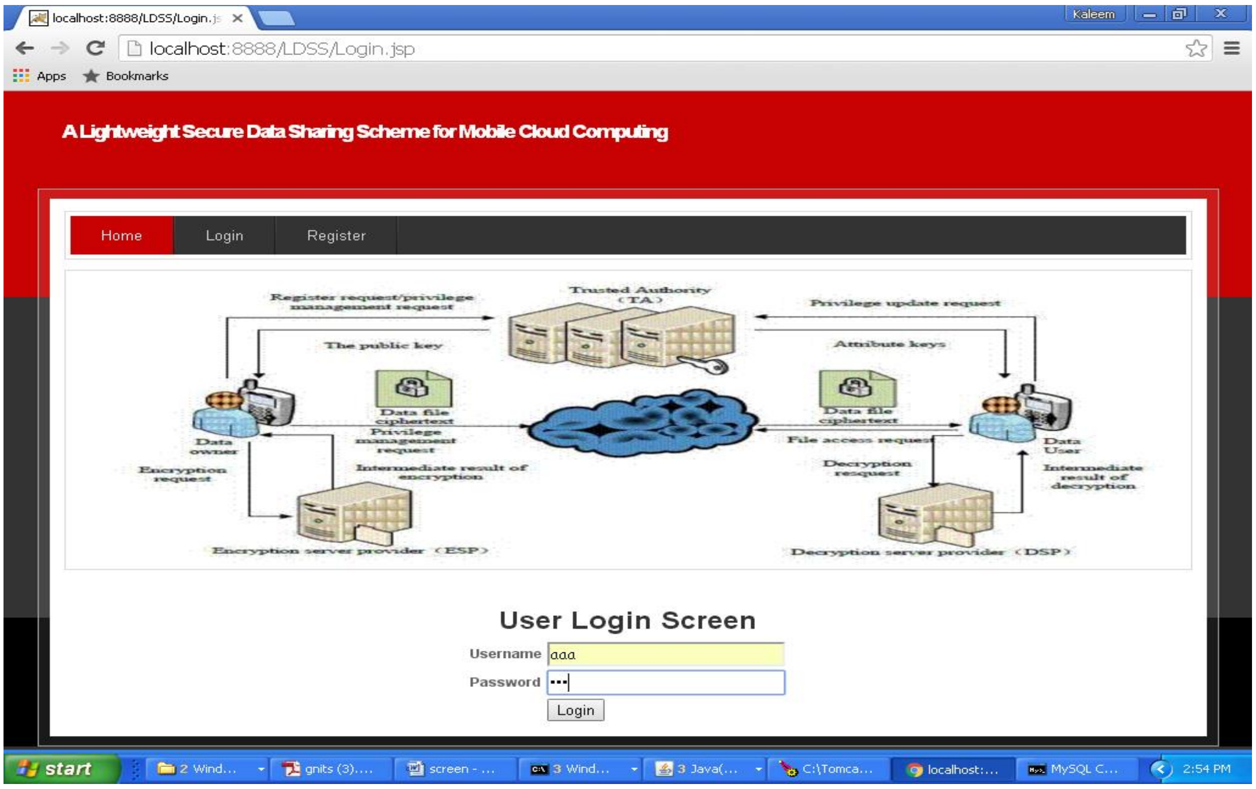
User Registration Screen

Username	ccc
Password	•••
Email	ccc@gmail.com
Contact No	9652861905
Address	hyd

Register

Figure 9.1.9: screenshot of registering other users

I added three users now login as one user and upload and share data


localhost:8888/LDSS/Login.jsp

A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing

Home Login Register

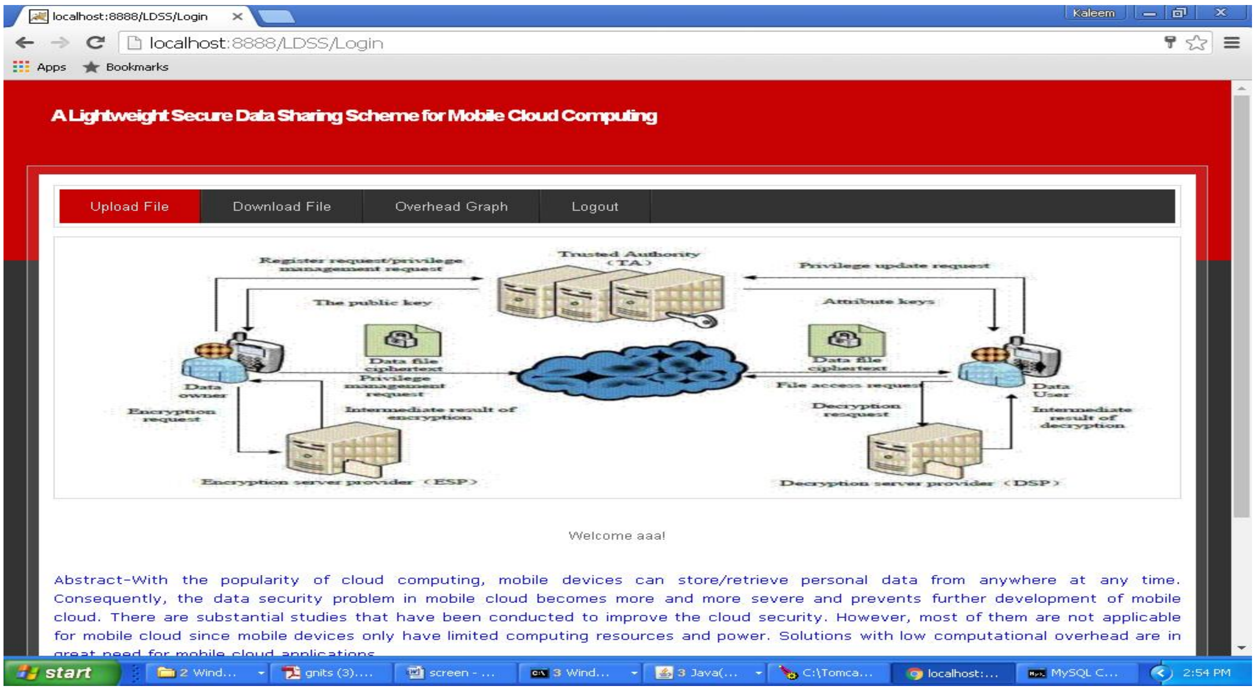
User Login Screen

Username	aaa
Password	•••

Login

Figure 9.1.10: Screenshots of user login

Will get below screen after successful login



Welcome aaa!

Abstract-With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power. Solutions with low computational overhead are in great need for mobile cloud applications.

Figure 9.1.11: After successful login

Click on 'Upload File' link to upload data

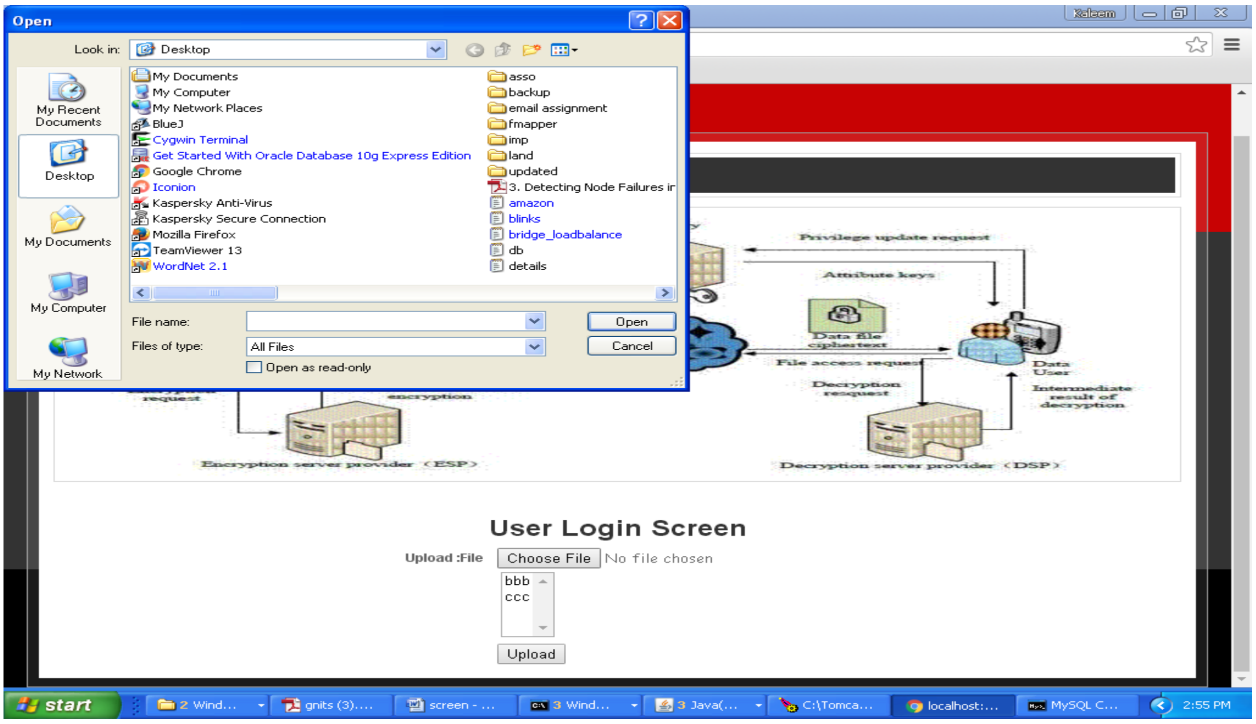


Figure 9.1.12: Screenshot of uploading data

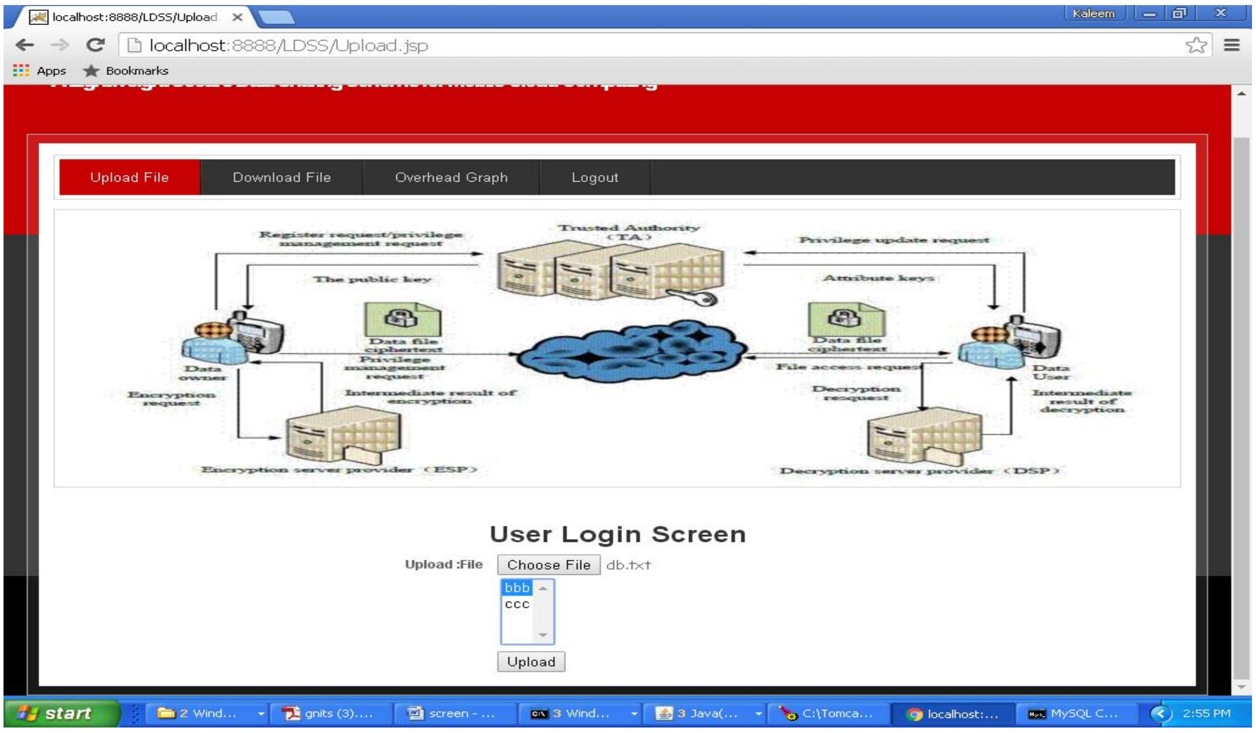


Figure 9.1.13: Screenshot of choosing file

In above screen user aaa upload 'db.txt' file and sharing with user bbb

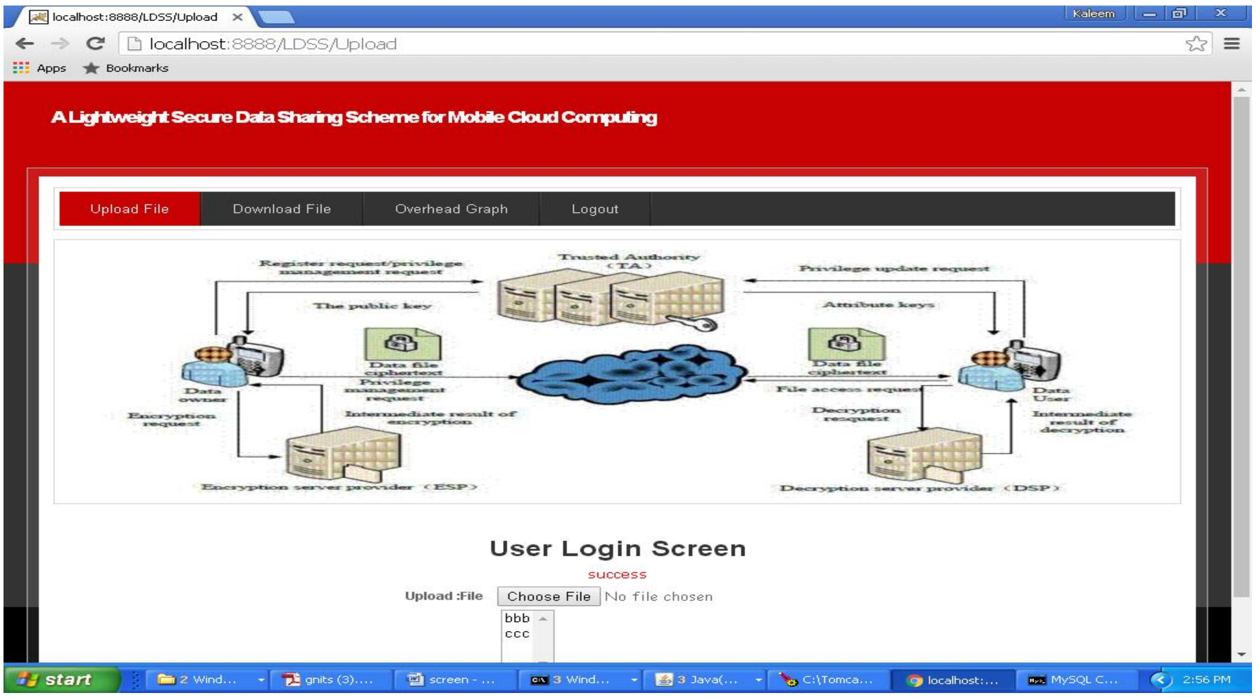
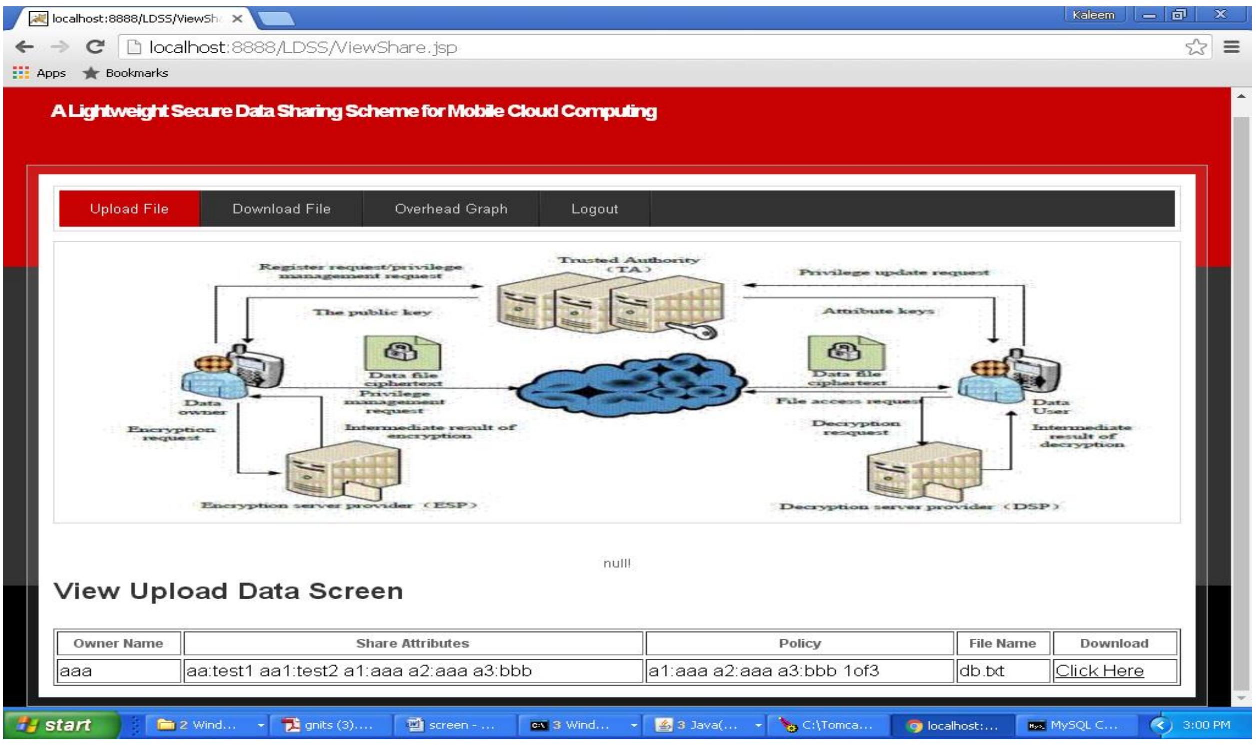


Figure 9.1.14: Screenshot after successfully uploading file

User aaa and user bbb can download the file but not user ccc. Now click on 'Download File' link to view share and to download file.

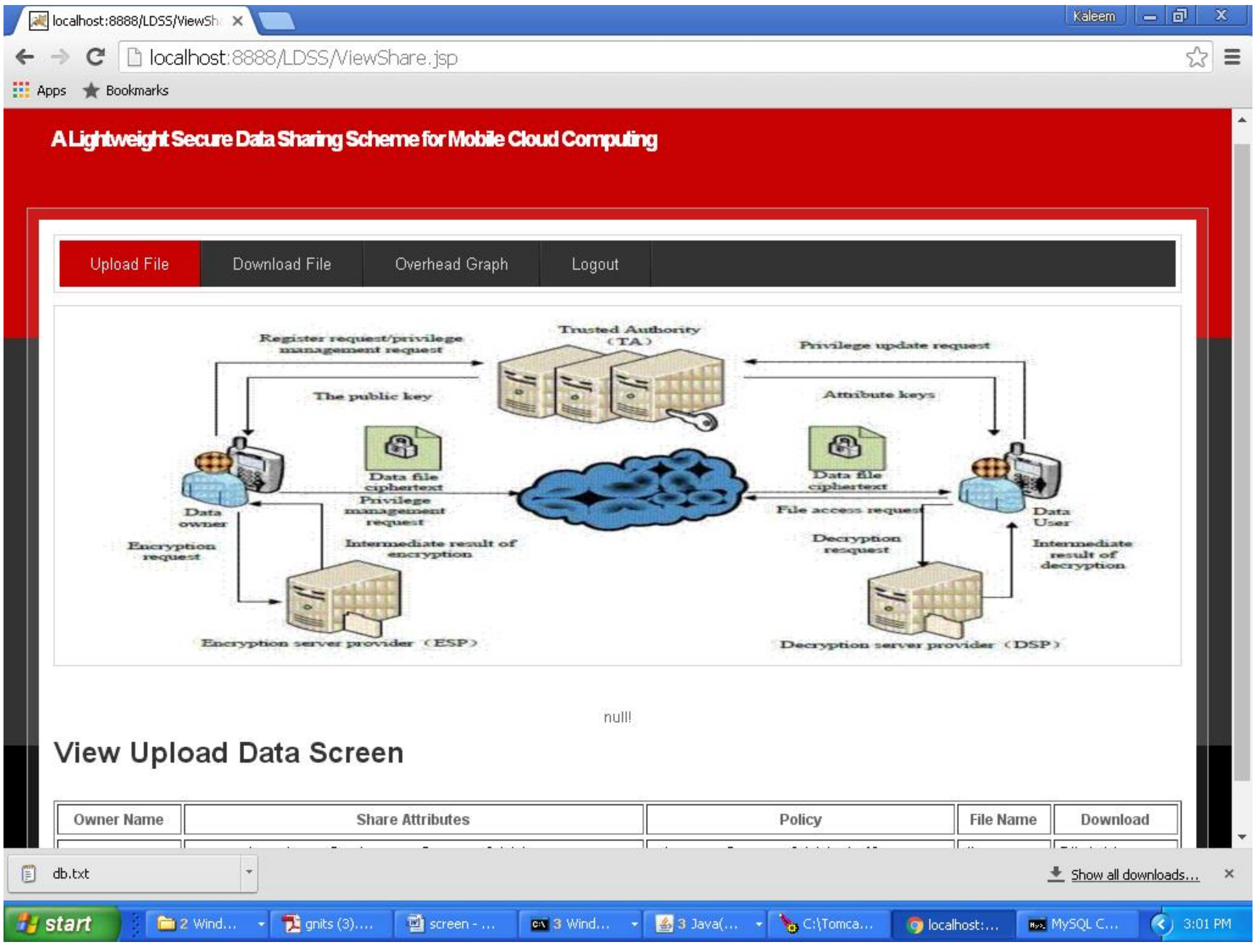

localhost:8888/LDSS/ViewShare.jsp
A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing
Upload File
Download File
Overhead Graph
Logout
Trusted Authority (TA)
Register request/privilege management request
Privilege update request
The public key
Attribute keys
Data file ciphertext
Data owner
Data User
Privilege management request
File access request
Encryption request
Intermediate result of encryption
Decryption request
Intermediate result of decryption
Encryption server provider (ESP)
Decryption server provider (DSP)
null!

View Upload Data Screen

Owner Name	Share Attributes	Policy	File Name	Download
aaa	aa:test1 aa1:test2 a1:aaa a2:aaa a3:bbb	a1:aaa a2:aaa a3:bbb 1of3	db.txt	Click Here

Figure 9.1.15: Screenshot of data screen

Now click on 'Click Here' to link to download file

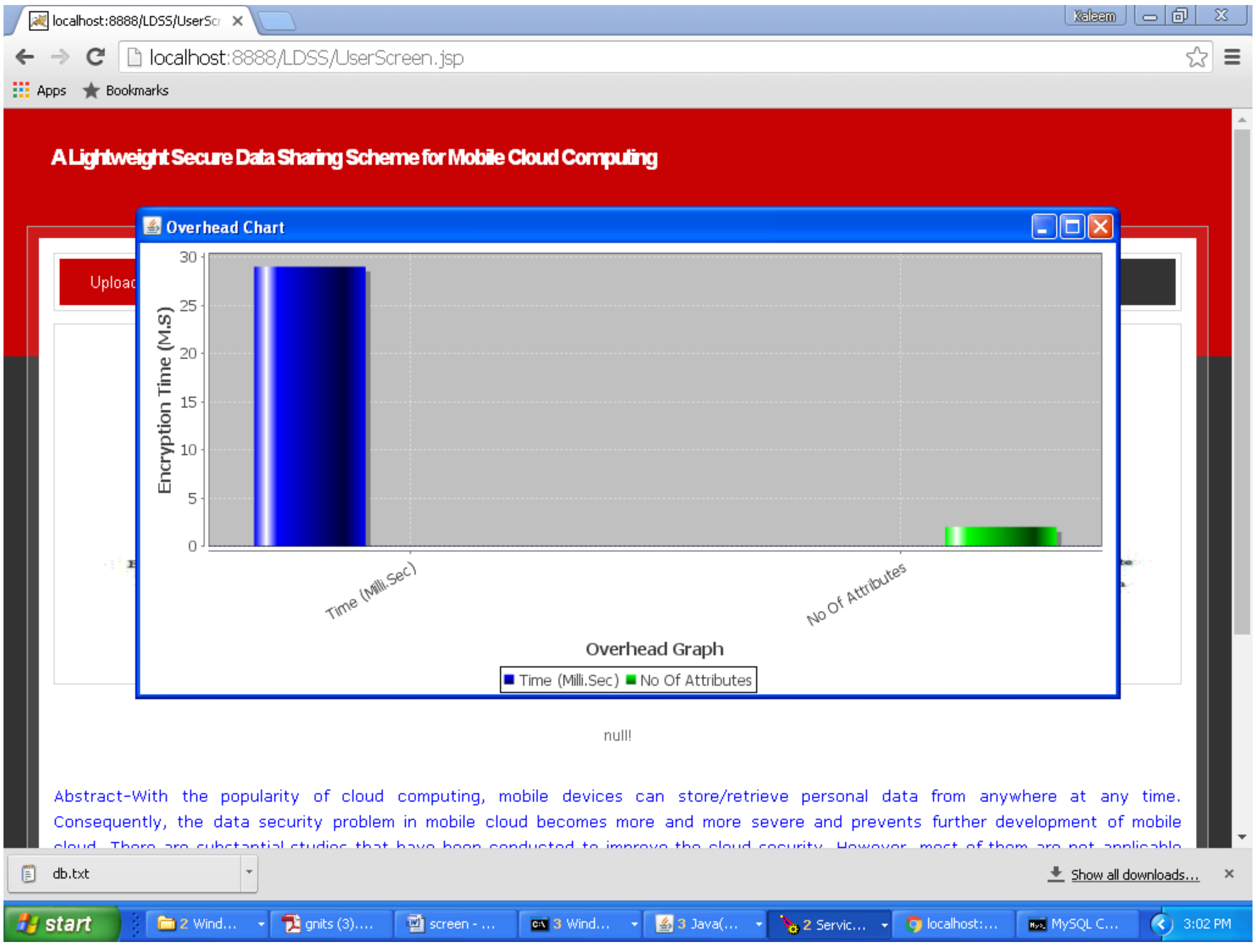

localhost:8888/LDSS/ViewShare.jsp
Apps
Bookmarks
A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing
Upload File
Download File
Overhead Graph
Logout
Trusted Authority (TA)
Register request/privilege management request
Privilege update request
The public key
Attribute keys
Data file ciphertext
Privilege management request
Data owner
Encryption request
Intermediate result of encryption
File access request
Decryption request
Data User
Intermediate result of decryption
Encryption server provider (ESP)
Decryption server provider (DSP)
null!
db.txt
Show all downloads...
start
3:01 PM

View Upload Data Screen

Owner Name	Share Attributes	Policy	File Name	Download

Figure 9.1.16: Screenshot of downloading the file

In browser task bar we can see db.txt file downloaded. Now click on ‘Overhead Graph link’ to generate graph


localhost:8888/LDSS/UserScreen.jsp
A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing
Overhead Chart
Encryption Time (M.S)
Time (Milli.Sec)
No Of Attributes
Overhead Graph
■ Time (Milli.Sec) ■ No Of Attributes
db.txt
Show all downloads...

null!

Abstract-With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud. There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable

Figure 9.1.17: Screenshot of graph generated

Now logout and login as ccc

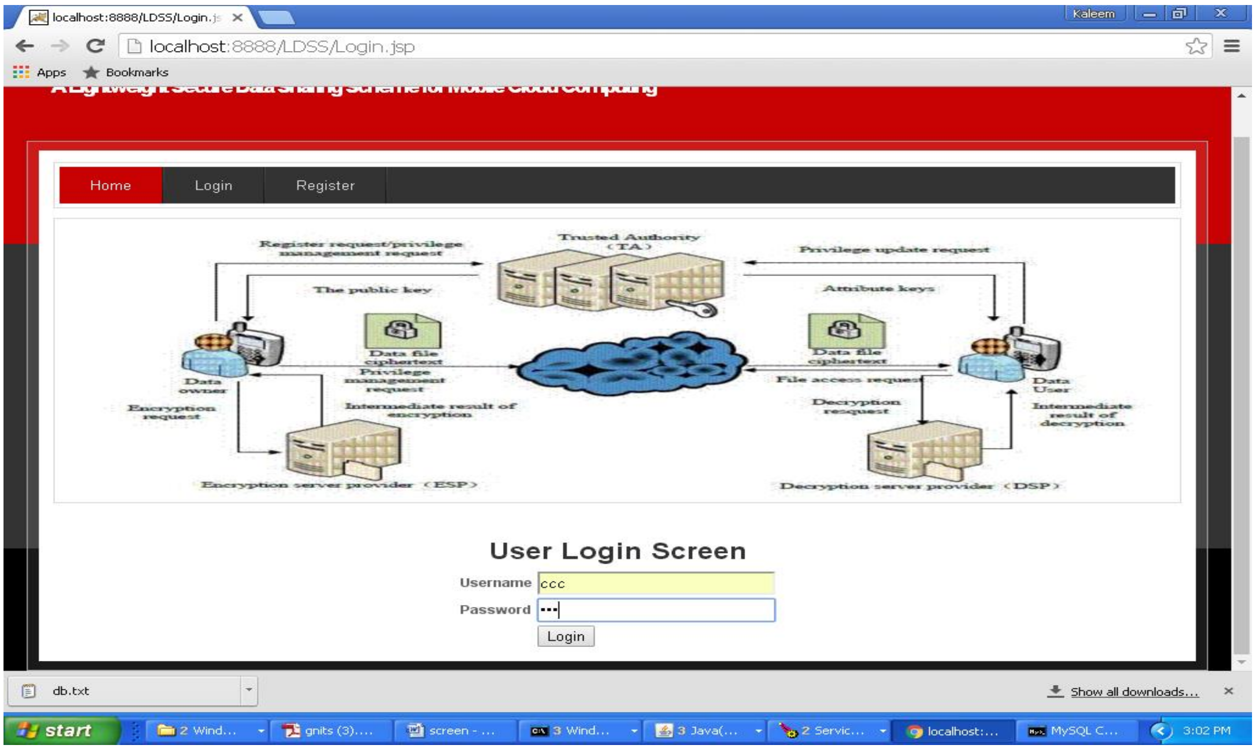
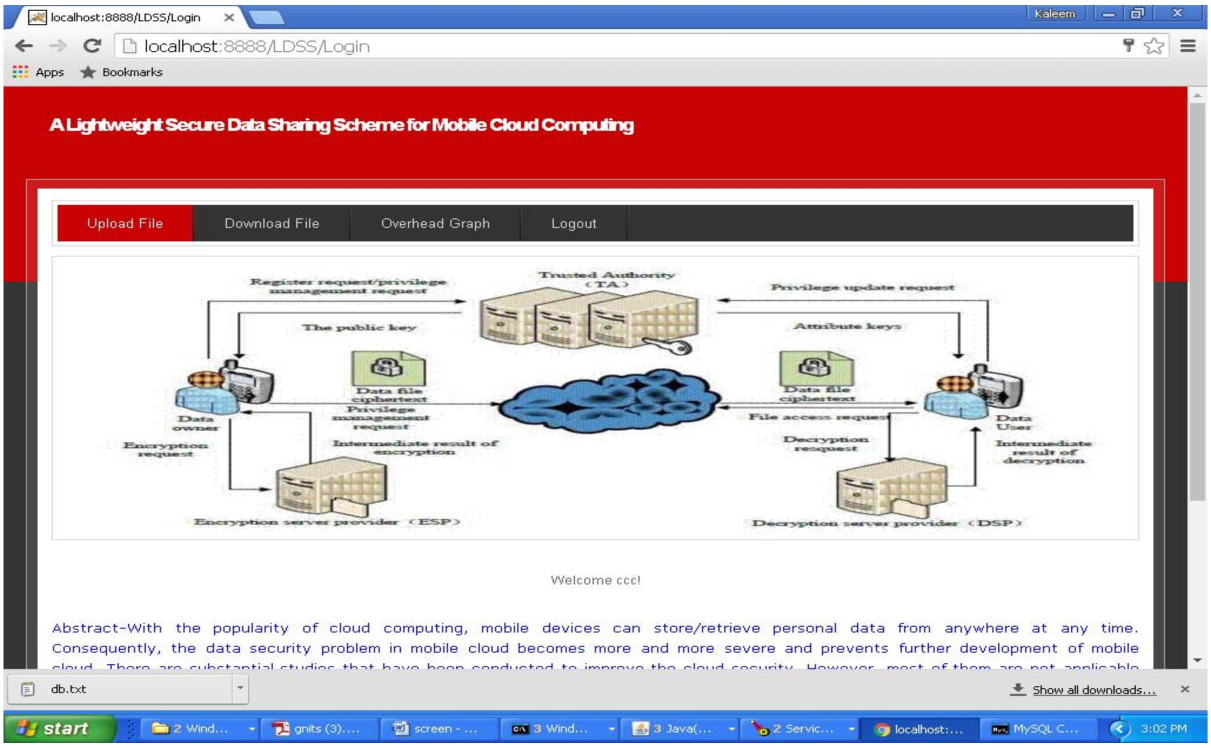


Figure 9.1.18: Screenshot of login of another user



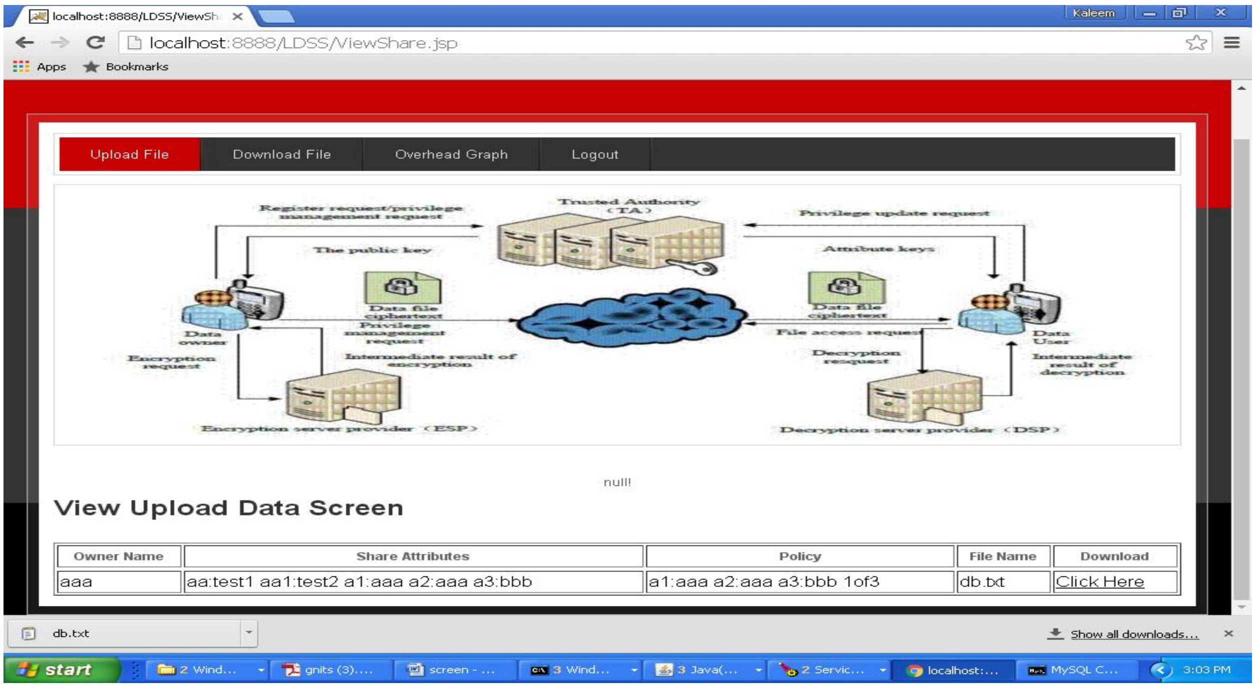
A Lightweight Secure Data Sharing Scheme for Mobile Cloud Computing

Welcome ccc!

Abstract-With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time. Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile

Figure 9.1.19: Screenshot of login of another user

Now click on 'Download File' link to get below screen



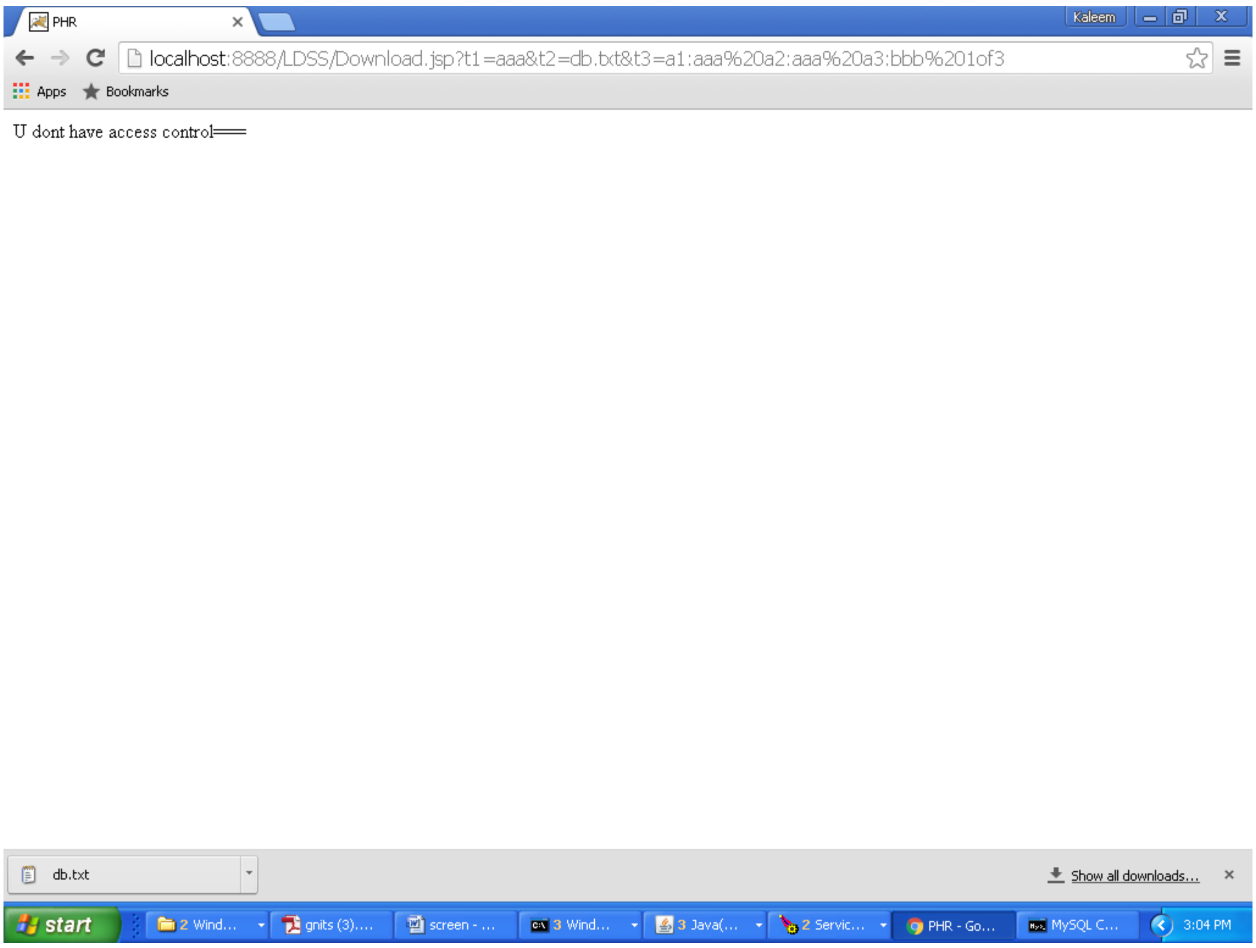
null!

View Upload Data Screen

Owner Name	Share Attributes	Policy	File Name	Download
aaa	aa:test1 aa1:test2 a1:aaa a2:aaa a3:bbb	a1:aaa a2:aaa a3:bbb 1of3	db.txt	Click Here

Figure 9.1.20: Screenshot of file downloaded by the selected person

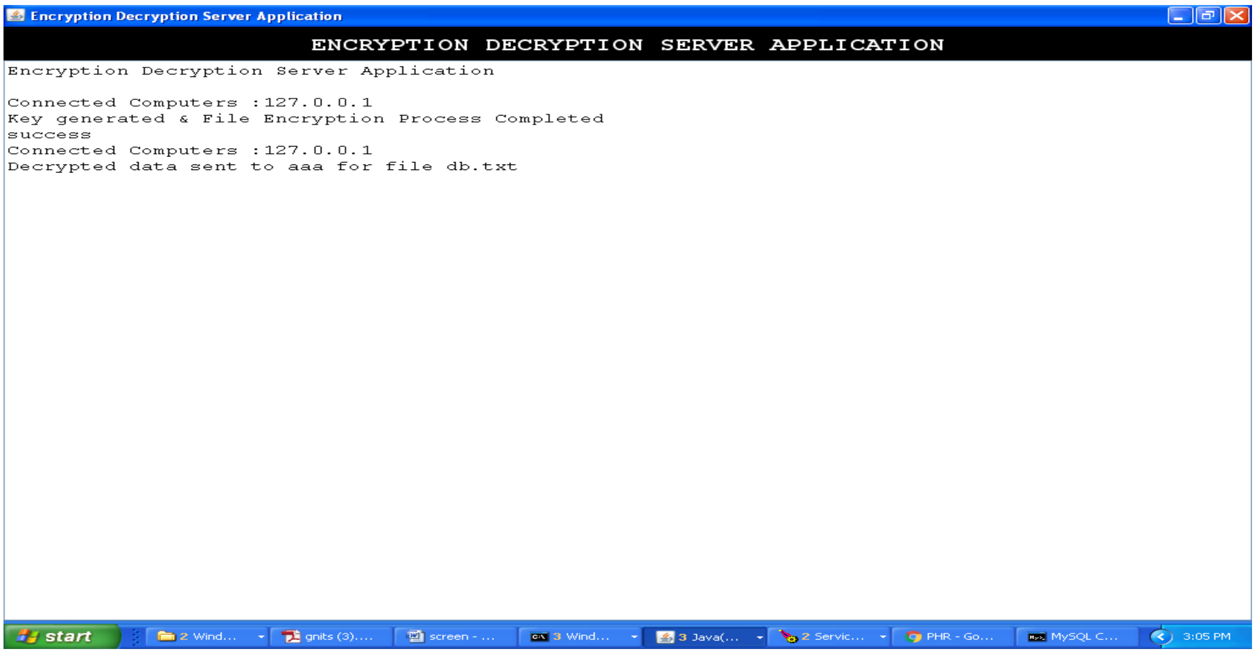
Now click on ‘Click Here’ link to see share data


PHR
localhost:8888/LDSS/Download.jsp?t1=aaa&t2=db.txt&t3=a1:aaa%20a2:aaa%20a3:bbb%201of3
Apps
Bookmarks
db.txt
Show all downloads...
start
3:04 PM

U dont have access control===

Figure 9.1.21: Screenshot of not having permission on the file

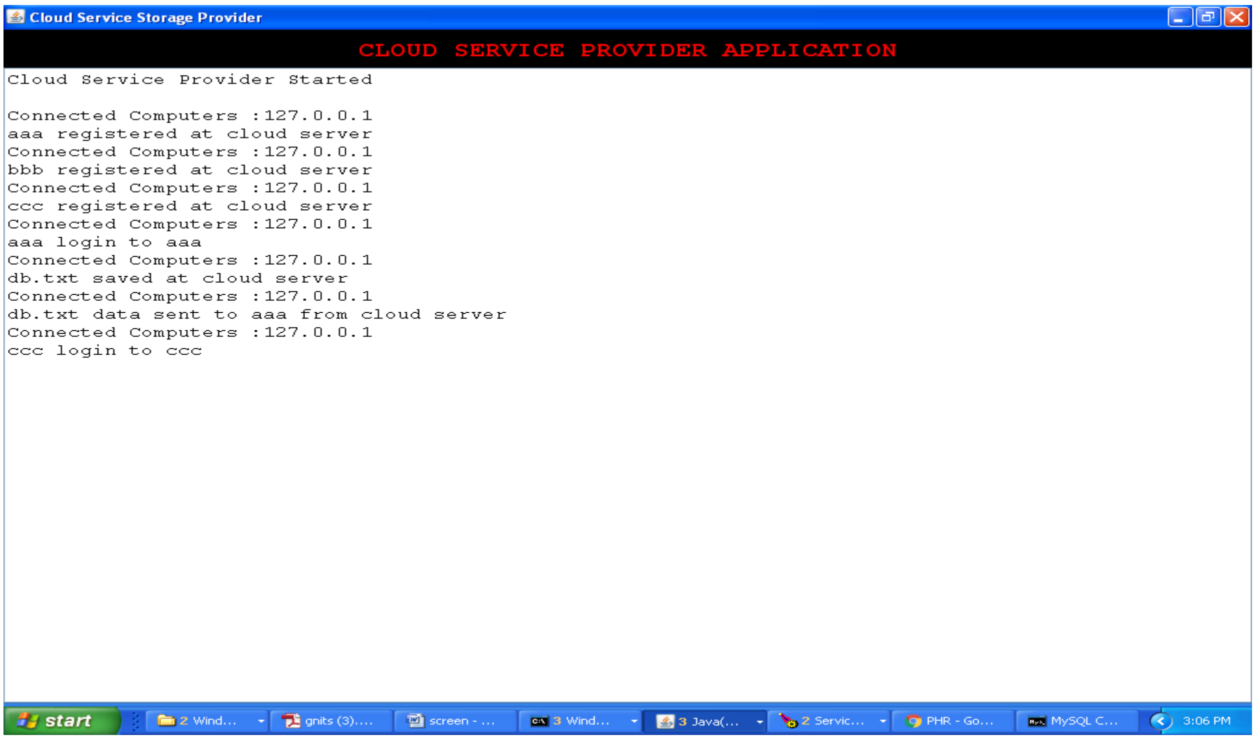
In above screen we can see user aaa has not given access to user ccc. In below screen we can see encryption and decryption server status


Encryption Decryption Server Application
ENCRYPTION DECRYPTION SERVER APPLICATION

```
Encryption Decryption Server Application

Connected Computers :127.0.0.1
Key generated & File Encryption Process Completed
success
Connected Computers :127.0.0.1
Decrypted data sent to aaa for file db.txt
```

Figure 9.1.22: Screenshot of status on encryption decryption screen

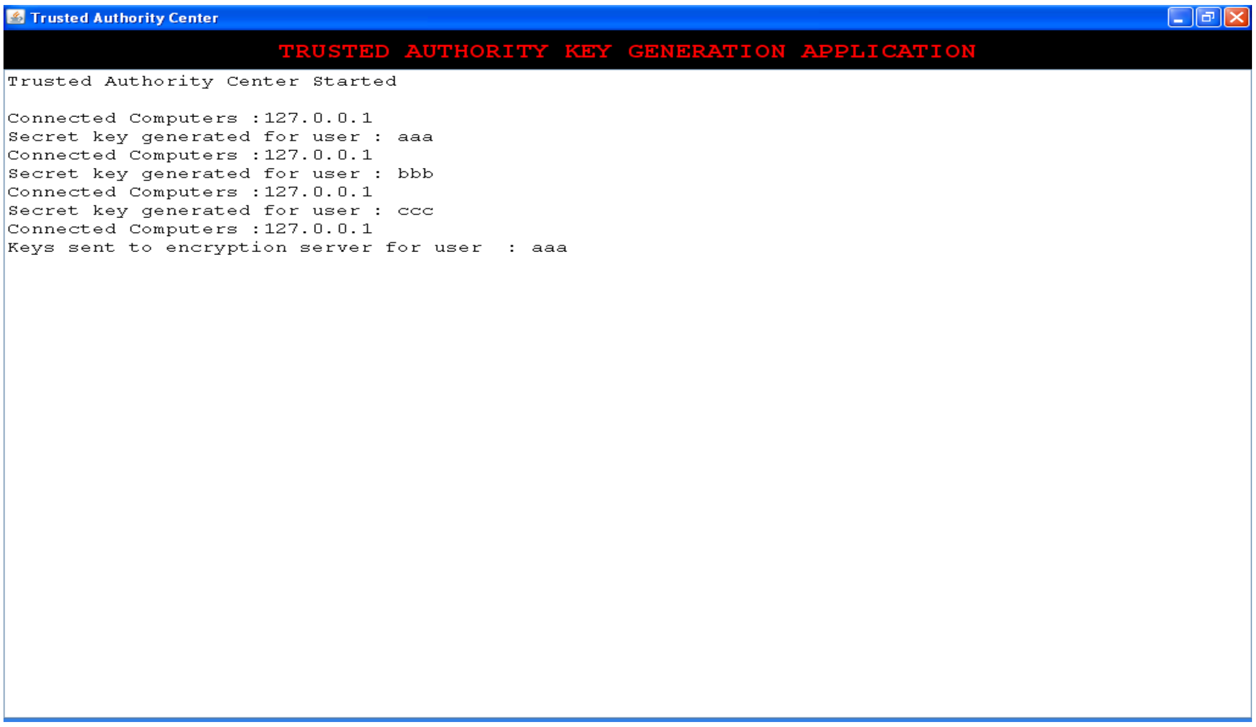
Cloud Server screen


Cloud Service Storage Provider
CLOUD SERVICE PROVIDER APPLICATION

```
Cloud Service Provider Started

Connected Computers :127.0.0.1
aaa registered at cloud server
Connected Computers :127.0.0.1
bbb registered at cloud server
Connected Computers :127.0.0.1
ccc registered at cloud server
Connected Computers :127.0.0.1
aaa login to aaa
Connected Computers :127.0.0.1
db.txt saved at cloud server
Connected Computers :127.0.0.1
db.txt data sent to aaa from cloud server
Connected Computers :127.0.0.1
ccc login to ccc
```

Figure 9.1.23: Screenshot of status on cloud server

Trusted Authority screen



Trusted Authority Center

TRUSTED AUTHORITY KEY GENERATION APPLICATION

```
Trusted Authority Center Started

Connected Computers :127.0.0.1
Secret key generated for user : aaa
Connected Computers :127.0.0.1
Secret key generated for user : bbb
Connected Computers :127.0.0.1
Secret key generated for user : ccc
Connected Computers :127.0.0.1
Keys sent to encryption server for user  : aaa
```

Figure 9.1.24: Screenshot of status of trusted authority server

Chapter 10

Conclusion and Future Scope

10.1 Conclusion

In recent years, many studies on access control in cloud are based on attribute-based encryption algorithm (ABE). However, traditional ABE is not suitable for mobile cloud because it is computationally intensive and mobile devices only have limited resources. In this paper, we propose LDSS to address this issue. It introduces a novel LDSS-CP-ABE algorithm to migrate major computation overhead from mobile devices onto proxy servers, thus it can solve the secure data sharing problem in mobile cloud. The experimental results show that LDSS can ensure data privacy in mobile cloud and reduce the overhead on users' side in mobile cloud.

10.2 Future Scope

In the future work, we will design new approaches to ensure data integrity. To further tap the potential of mobile cloud, we will also study how to do ciphertext retrieval over existing data sharing schemes.

References

[1] Ruixuan Li; et.al. 2017 Implementing secure scheme for cloud computing. in: Attribute-based fine-grained access control with efficient revocation in cloud storage systems. ASIACCS 2013, pp. 523-528, 2013.

[2] Gentry C, Halevi S. Implementing gentry's fully-homomorphic encryption scheme. in: Advances in Cryptology–EUROCRYPT 2011. Berlin, Heidelberg: Springer press, pp. 129-148, 2011.

[3] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. in: Proceeding of IEEE Symposium on Foundations of Computer Science. California, USA: IEEE press, pp. 97-106, Oct. 2011.

[4] Qihua Wang, Hongxia Jin. "Data leakage mitigation for discretionary access control in collaboration clouds". the 16th ACM Symposium on Access Control Models and Technologies (SACMAT), pp.103-122, Jun. 2011.

[5] Adam Skillen and Mohammad Mannan. On Implementing Deniable Storage Encryption for Mobile Devices. the 20th Annual Network and Distributed System Security Symposium (NDSS), Feb. 2013.

[6] Wang W, Li Z, Owens R, et al. Secure and efficient access to outsourced data. in: Proceedings of the 2009 ACM workshop on Cloud computing security. Chicago, USA: ACM pp. 55-66, 2009.

[7] Maheshwari U, Vingralek R, Shapiro W. How to build a trusted database system on untrusted storage. in: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4. USENIX Association, pp. 10-12, 2000.

[8] Crampton J, Martin K, Wild P. On key assignment for hierarchical access control. in: Computer Security Foundations Workshop. IEEE press, pp. 14-111, 2006.

[9] Shi E, Bethencourt J, Chan T H H, et al. Multi-dimensional range query over encrypted data. in: Proceedings of Symposium on Security and Privacy (SP), IEEE press, 2007. 350364

[10] Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje Urs. Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data. IEEE INFOCOM 2012, Orlando, Florida, March 25-30, 2012