INDEX

[CHAPTER 1 1](#_Toc448574546)

[INTRODUCTION 1](#_Toc448574547)

[1.1 Group Signature: 4](#_Toc448574548)

[1.2 Dynamic Broadcast Encryption : 4](#_Toc448574549)

[1.3 Technoogies used 5](#_Toc448574550)

[1.3.1 Introduction To Java: 5](#_Toc448574551)

[1.3.2 The Java Platform: 7](#_Toc448574552)

[1.3.3 Working Of Java: 8](#_Toc448574553)

[1.3.4 Swing: 9](#_Toc448574554)

[1.3.5 Benefits of Swing: 9](#_Toc448574555)

[1.4 Domain Description: 10](#_Toc448574556)

[1.4.1 Data Mining Uses: 11](#_Toc448574557)

[CHAPTER 2 13](#_Toc448574558)

[SYSTEM ANALYSIS 13](#_Toc448574559)

[2.1 Existing System : 13](#_Toc448574560)

[Disadvantage: 13](#_Toc448574561)

[2.2 Proposed system: 13](#_Toc448574562)

[2.3 System Configuration:- 15](#_Toc448574563)

[2.3.1 Hardware configuration:- 15](#_Toc448574564)

[2.3.2 Software configuration:- 16](#_Toc448574565)

[2.4 Functional and Non-Functional Requirements: 16](#_Toc448574566)

[2.4.1 Functoinal Requirements: 16](#_Toc448574567)

[2.4.2 Non Functional Requirements 17](#_Toc448574571)

[CHAPTER 3 19](#_Toc448574580)

[SYTEM DESIGN 19](#_Toc448574581)

[3.1 System Model 19](#_Toc448574582)

[3.2 Design Goals 20](#_Toc448574583)

[3.2.1 Access control: 20](#_Toc448574584)

[3.2.2 Data confidentiality: 20](#_Toc448574585)

[3.3.3 Anonymity and traceability: 20](#_Toc448574586)

[3.3.4 Efficiency: 21](#_Toc448574587)

[3.5 UML: 21](#_Toc448574588)

[3.6 UML Diagrams: 21](#_Toc448574589)

[3.6.1 UML is a language: 22](#_Toc448574590)

[3.6.2 UML Specifying: 22](#_Toc448574591)

[3.6.3 UML Visualization: 22](#_Toc448574592)

[3.6.4 UML Constructing: 22](#_Toc448574593)

[3.6.5 UML Documenting: 22](#_Toc448574594)

[3.6.6 Uses of UML: 23](#_Toc448574595)

[3.6.7 Building blocks of UML: 24](#_Toc448574596)

[3.7 Use-Case diagram: 25](#_Toc448574602)

[3.8 Class Diagram: 26](#_Toc448574603)

[3.9 Sequence Diagram 27](#_Toc448574604)

[3.10 Collaboration diagram 28](#_Toc448574605)

[3.11 State machine diagrams 29](#_Toc448574606)

[3.12 Activity Diagram: 30](#_Toc448574607)

[3.12.1 How to Draw: Activity Diagrams 30](#_Toc448574608)

[3.12.2 When to Use: Activity Diagrams 30](#_Toc448574609)

[3.13 Component diagram 31](#_Toc448574610)

[3.14 Deployment Diagram 32](#_Toc448574611)

[CHAPTER 4 33](#_Toc448574612)

[SYSTEM IMPLEMENTATION 33](#_Toc448574613)

[4.1 Modules : 33](#_Toc448574614)

[4.1.1 Registration: 33](#_Toc448574615)

[4.1.2 Login: 33](#_Toc448574616)

[4.1.3 File Upload: 33](#_Toc448574617)

[4.1.4 Chart Creation: 33](#_Toc448574618)

[4.1.5 File Download: 33](#_Toc448574619)

[4.1.6 User Deletion: 34](#_Toc448574620)

[CHAPTER 5 35](#_Toc448574621)

[SCREEN SHOTS 35](#_Toc448574622)

[5.1 Output Screens 35](#_Toc448574623)

[5.2 Coding 43](#_Toc448574624)

[CHAPTER 6 55](#_Toc448574625)

[TESTING 55](#_Toc448574626)

[6.1 Types of tests 56](#_Toc448574627)

[6.1.1 Unit testing 56](#_Toc448574628)

[6.1.2 Integration testing 56](#_Toc448574629)

[6.1.3 Acceptance testing 57](#_Toc448574630)

[6.1.4 Functional test 57](#_Toc448574631)

[6.1.5 System Test 58](#_Toc448574632)

[6.1.6 White Box Testing 58](#_Toc448574633)

[6.1.7 Black Box Testing 58](#_Toc448574634)

[6.2 Testing strategy 58](#_Toc448574635)

[6.3 Test objectives 59](#_Toc448574636)

[6.4 Features to be tested 59](#_Toc448574637)

[6.5 Test Results: 59](#_Toc448574638)

[CONCLUSION 60](#_Toc448574639)

[REFERENCES 61](#_Toc448574640)

# CHAPTER 1

# INTRODUCTION

Cloud computing is recognized as an alternative to traditional information technology due to its intrinsic resource-sharing and low-maintenance characteristics. In cloud computing, the cloud service providers (CSPs), such as Amazon, are able to deliver various services to cloud users with the help of powerful datacenters. By migrating the local data management systems into cloud servers, users can enjoy high-quality services and save significant investments on their local infrastructures. One of the most fundamental services offered by cloud providers is data storage. Let us consider a practical data application. A company allows its staffs in the same group or department to store and share files in the cloud. By utilizing the cloud, the staffs can be completely released from the troublesome local data storage and maintenance. However, it also poses a significant risk to the confidentiality of those stored files. Specifically, the cloud servers managed by cloud providers are not fully trusted by users while the data files stored in the cloud may be sensitive and confidential, such as business plans. To preserve data privacy, a basic solution is to encrypt data files, and then upload the encrypted data into the cloud.

Unfortunately, designing an efficient and secure data sharing scheme for groups in the cloud is not an easy task due to the following challenging issues. First, identity privacy is one of the most significant obstacles for the wide deployment of cloud computing. Without the guarantee of identity privacy, users may be unwilling to join in cloud computing systems because their real identities could be easily disclosed to cloud providers and attackers. On the other hand, unconditional identity privacy may incur the abuse of privacy. For example, a misbehaved staff can deceive others in the company by sharing false files without being traceable. Therefore, traceability, which enables the group manager (e.g., a company manager) to reveal the real identity of a user, is also highly desirable. Second, it is highly recommended that any member in a group should be able to fully enjoy the data storing and sharing services provided by the cloud, which is defined as the multiple-owner manner.

Compared with the single-owner manner, where only the group manager can store and modify data in the cloud, the multiple-owner manner is more flexible in practical applications. More concretely, each user in the group is able to not only read data, but also modify his/ her part of data in the entire data file shared by the company.

Last but not least, groups are normally dynamic in practice, e.g., new staff participation and current employee revocation in a company. The changes of membership make secure data sharing extremely difficult. On one hand, the anonymous system challenges new granted users to learn the content of data files stored before their participation, because it is impossible for new granted users to contact with anonymous data owners, and obtain the corresponding decryption keys. On the other hand, an efficient membership revocation mechanism without updating the secret keys of the remaining users is also desired to minimize the complexity of key management. Several security schemes for data sharing on untrusted servers have been proposed. In these approaches, data owners store the encrypted data files in untrusted storage and distribute the corresponding decryption keys only to authorized users. Thus, unauthorized users as well as storage servers cannot learn the content of the data files because they have no knowledge of the decryption keys.

However, the complexities of user participation and revocation in these schemes are linearly increasing with the number of data owners and the number of revoked users, respectively. By setting a group with a single attribute, Lu et al. proposed a secure provenance scheme based on the ciphertext-policy attribute-based encryption technique, which allows any member in a group to share data with others. However, the issue of user revocation is not addressed in their scheme. Yu et al. presented a scalable and fine-grained data access control scheme in cloud computing based on the key policy attribute-based encryption (KP-ABE) technique Unfortunately, the singleowner manner hinders the adoption of their scheme into the case, where any user is granted to store and share data. our contributions. To solve the challenges presented above, we propose Mona, a secure multi-owner data sharing scheme for dynamic groups in the cloud. The main contributions of this paper include:

1. We propose a secure multi-owner data sharing scheme. It implies that any user in the group can securely share data with others by the untrusted cloud.
2. Our proposed scheme is able to support dynamic groups efficiently. Specifically, new granted users can directly decrypt data files uploaded before their participation without contacting with data owners. User revocation can be easily achieved through a novel revocation list without updating the secret keys of the remaining users. The size and computation overhead of encryption are constant and independent with the number of revoked users.
3. We provide secure and privacy-preserving access control to users, which guarantees any member in a group to anonymously utilize the cloud resource. Moreover, the real identities of data owners can be revealed by the group manager when disputes occur.
4. We provide rigorous security analysis, and perform extensive simulations to demonstrate the efficiency of our scheme in terms of storage and computation overhead.

## 1.1 Group Signature:

The concept of group signatures was first introduced in by Chaum and van Heyst. In general, a group signature scheme allows any member of the group to sign messages while keeping the identity secret from verifiers. Besides, the designated group manager can reveal the identity of the signature’s originator when a dispute occurs, which is denoted as traceability. In this paper, a variant of the short group signature scheme will be used to achieve anonymous access control, as it supports efficient membership revocation.

## 1.2 Dynamic Broadcast Encryption :

Broadcast encryption enables a broadcaster to transmit encrypted data to a set of users so that only a privileged subset of users can decrypt the data. Besides the above characteristics, dynamic broadcast encryption also allows the group manager to dynamically include new members while preserving previously computed information, i.e., user decryption keys need not be recomputed, the morphology and size of ciphertexts are unchanged and the group encryption key requires no modification. The first formal definition and construction of dynamic broadcast encryption are introduced based on the bilinear pairing technique in, which will be used as the basis for file sharing in dynamic groups.

## 1.3 Technoogies used

### 1.3.1 Introduction To Java:

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

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

Around the time the First Person project was floundering in consumer electronics, a new craze was gaining momentum in America; the craze was called "Web surfing." The World Wide Web, a name applied to the Internet's millions of linked HTML documents was suddenly becoming popular for use by the masses. The reason for this was the introduction of a graphical Web browser called Mosaic, developed by ncSA. The browser simplified Web browsing by combining text and graphics into a single interface to eliminate the need for users to learn many confusing UNIX and DOS commands. Navigating around the Web was much easier using Mosaic.

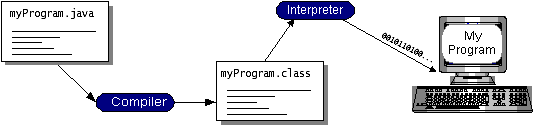
It has only been since 1994 that Oak technology has been applied to the Web. In 1994, two Sun developers created the first version of Hot Java, and then called Web Runner, which is a graphical browser for the Web that exists today. The browser was coded entirely in the Oak language, by this time called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

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

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* Simple
* Architecture neutral
* Object oriented
* Portable
* Distributed
* High performance
* Interpreted
* Multithreaded
* Robust
* Dynamic
* Secure

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



**Fig 1.1: Working Of Java**

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

### 1.3.2 The Java Platform:

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

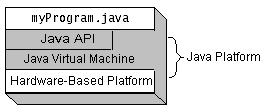
The Java platform has two components:

1. The java virtual mechine (Java VM)
2. The java application programming interface (Java API)

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

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

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



**Fig 1.2: The Java Platform**

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time bytecode compilers can bring performance close to that of native code without threatening portability.

### 1.3.3 Working Of Java:

For those who are new to object-oriented programming, the concept of a class will be new to you. Simplistically, a class is the definition for a segment of code that can contain both data and functions. When the interpreter executes a class, it looks for a particular method by the name of **main,** which will sound familiar to C programmers. The main method is passed as a parameter an array of strings (similar to the argv[] of C), and is declared as a static method.

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

### 1.3.4 Swing:

Swing contains all the components. It’s a big library, but it’s designed to have appropriate complexity for the task at hand – if something is simple, you don’t have to write much code but as you try to do more your code becomes increasingly complex. This means an easy entry point, but you’ve got the power if you need it.

Swing has great depth. This section does not attempt to be comprehensive, but instead introduces the power and simplicity of Swing to get you started using the library. Please be aware that what you see here is intended to be simple. If you need to do more, then Swing can probably give you what you want if you’re willing to do the research by hunting through the online documentation from Sun.

### 1.3.5 Benefits of Swing:

Swing components are Beans, so they can be used in any development environment that supports Beans. Swing provides a full set of UI components. For speed, all the components are lightweight and Swing is written entirely in Java for portability.

Swing could be called “orthogonality of use;” that is, once you pick up the general ideas about the library you can apply them everywhere. Primarily because of the Beans naming conventions.

Keyboard navigation is automatic – you can use a Swing application without the mouse, but you don’t have to do any extra programming. Scrolling support is effortless – you simply wrap your component in a JScrollPane as you add it to your form. Other features such as tool tips typically require a single line of code to implement.

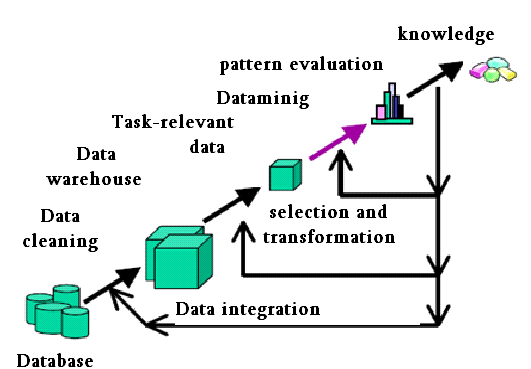
Swing also supports something called “pluggable look and feel,” which means that the appearance of the UI can be dynamically changed to suit the expectations of users working under different platforms and operating systems. It’s even possible to invent your own look and feel.

## 

## 1.4 Domain Description:

Data mining involves the use of sophisticated data analysis tools to discover previously unknown, valid patterns and relationships in large data sets. These tools can include statistical models, mathematical algorithms, and machine learning methods (algorithms that improve their performance automatically through experience, such as neural networks or decision trees). Consequently, data mining consists of more than collecting and managing data, it also includes analysis and prediction.

Data mining can be performed on data represented in quantitative, textual, or multimedia forms. Data mining applications can use a variety of parameters to examine the data. They include association (patterns where one event is connected to another event, such as purchasing a pen and purchasing paper), sequence or path analysis (patterns where one event leads to another event, such as the birth of a child and purchasing diapers), classification (identification of new patterns, such as coincidences between duct tape purchases and plastic sheeting purchases), clustering (finding and visually documenting groups of previously unknown facts, such as geographic location and brand preferences), and forecasting (discovering patterns from which one can make reasonable predictions regarding future activities, such as the prediction that people who join an athletic club may take exercise classes)

 **Fig 1.3 knowledge discovery process**

### 1.4.1 Data Mining Uses:

Data mining is used for a variety of purposes in both the private and public sectors.

* Industries such as banking, insurance, medicine, and retailing commonly use data mining to reduce costs, enhance research, and increase sales. For example, the insurance and banking industries use data mining applications to detect fraud and assist in risk assessment (e.g., credit scoring).
* Using customer data collected over several years, companies can develop models that predict whether a customer is a good credit risk, or whether an accident claim may be fraudulent and should be investigated more closely.
* The medical community sometimes uses data mining to help predict the effectiveness of a procedure or medicine.
* Pharmaceutical firms use data mining of chemical compounds and genetic material to help guide research on new treatments for diseases.

Retailers can use information collected through affinity programs (e.g., shoppers’ club cards, frequent flyer points, contests) to assess the effectiveness of product selection and placement decisions, coupon offers, and which products are often purchased together.

# CHAPTER 2

# SYSTEM ANALYSIS

## 2.1 Existing System :

Several security schemes for data sharing on un trusted servers have been proposed. In these approaches, data owners store the encrypted data files in un trusted storage and distribute the corresponding decryption keys only to authorized users. Thus, unauthorized users as well as storage servers cannot learn the content of the data files because they have no knowledge of the decryption keys. However, the complexities of user participation and revocation in these schemes are linearly increasing with the number of data owners and the number of revoked users, respectively. By setting a group with a single attribute, Lu et al. Proposed a secure provenance scheme based on the cipher text policy attribute-based encryption technique, which allows any member in a group to share data with others. However, the issue of user revocation is not addressed in their scheme. Yu et al. presented a scalable and fine-grained data access control scheme in cloud computing based on the key policy attribute-based encryption (KP-ABE) technique. Unfortunately, the single-owner manner hinders the adoption of their scheme into the case, where any user is granted to store and share data.

### Disadvantage:

In these Systems there is no user revocation and Security is less than our proposed system.

## 2.2 Proposed system:

Our contributions. To solve the challenges presented above, we propose Mona, a secure multi-owner data sharing scheme for dynamic groups in the cloud. The main contributions of this paper include:

1. We propose a secure multi-owner data sharing scheme. It implies that any user in the group can securely share data with others by the untrusted cloud.

2. Our proposed scheme is able to support dynamic groups efficiently. Specifically, new granted users can directly decrypt data files uploaded before their participation without contacting with data owners.

User revocation can be easily achieved through a novel revocation list without updating the secret keys of the remaining users. The size and computation overhead of encryption are constant and independent with the number of revoked users.

3. We provide secure and privacy-preserving access control to users, which guarantees any member in a group to anonymously utilize the cloud resource. Moreover, the real identities of data owners can be revealed by the group manager when disputes occur.

4. We provide rigorous security analysis, and per-form extensive simulations to demonstrate the efficiency of our scheme in terms of storage and computation overhead.

To achieve secure data sharing for dynamic groups in the cloud, we expect to combine the group signature and dynamic broadcast encryption techniques. Specially, the group signature scheme enables users to anonymously use the cloud resources, and the dynamic broadcast encryption technique allows data owners to securely share their data files with others including new joining users.

Unfortunately, each user has to compute revocation parameters to protect the confidentiality from the revoked users in the dynamic broadcast encryption scheme, which results in that both the computation overhead of the encryption and the size of the ciphertext increase with the number of revoked users. Thus, the heavy overhead and large ciphertext size may hinder the adoption of the broadcast encryption scheme to capacity-limited users.

To tackle this challenging issue, we let the group manager compute the revocation parameters and make the result public available by migrating them into the cloud. Such a design can significantly reduce the computation overhead of users to encrypt files and the ciphertext size. Specially, the computation overhead of users for encryption operations and the ciphertext size are constant and independent of the revocation users.

From the above analysis, we can observe that how to securely share data files in a multiple-owner manner for dynamic groups while preserving identity privacy from an untrusted cloud remains to be a challenging issue. In this paper, we propose a novel Mona protocol for secure data sharing in cloud computing. Compared with the existing works, Mona offers unique features as follows:

1. Any user in the group can store and share data files with others by the cloud.

2. The encryption complexity and size of ciphertexts are independent with the number of revoked users in the system.

3. User revocation can be achieved without updating the private keys of the remaining users.

4. A new user can directly decrypt the files stored in the cloud before his participation.

## 2.3 System Configuration:-

### 2.3.1 Hardware configuration:-

* 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

### 2.3.2 Software configuration:-

* Operating System : Windows XP
* Programming Language : JAVA
* Java Version : JDK 1.6 & above.

## 2.4 Functional and Non-Functional Requirements:

### 2.4.1 Functoinal Requirements:

### Inputs:

In this we will enter ip address for client system. In practice, the mechanism can be executed either by the end-user or by a centralized server. In the former case, the source node is responsible for collecting cost reports from individual SRs and computing the path vector and payments. Such an implementation is suitable for an end-user of sufficient power computation capability. A centralized server implementation is more appropriate for resource-constrained users.

### Operation:

Routers are small physical devices that join multiple networks together. Technically, a router is a Layer3 gateway device, meaning that it connects two or more networks and that the router operates at the network layer of the OSI model. This module is used for removing the Hackers among the nodes. This module consists of a textbox field which is used to enter the destination IP address. After that select the node. Now click on start button in server module. With this the data is transferred to the destination through the selected node. To avoid jamming in that node click on particular avoid jammer node. If we want to know the time elapsed for sending the file click on Time elapsed button in Router .Now go to client module there you will get a message saying that file received.Defination of Hackers: a person who illegally gains access to and sometimes tampers with information in a computer system

### Output:

By exaggerating its capacity, node may relay a larger fraction of traffic and receive a higher payment, leading to a possibly higher profit. To maintain the overall truthfulness of the mechanism, misreporting of capacity information should be eliminated by design.

### 2.4.2 Non Functional Requirements

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of users of the existing system to give the requirement specifications because they are the people who finally use the system.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system.
* The system should be accurate.
* Te system should be better than existing system.

### Features:

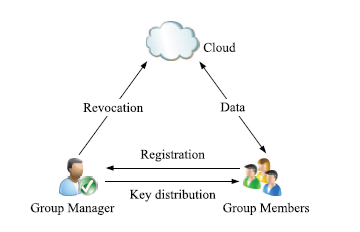
* Portability:It should run on specified platforms successfully. To achieve this we should test the product on all platforms before launching the product. If our project runs successfully on different platforms then our system is portable in nature.
* Reliability:The system should perform its intended functions under specified conditions. If our system satisfies all the specified conditions then it is Reliable in nature.
* Reusability:The system should be extremely reusable as a whole or part. Make the system modularize and make sure that modules are loosely coupled. This project is having reusability nature because we can reuse whole or part of this project on other systems.
* Robustness:The system on the whole should be robust enough to perform well under different circumstances without any inconsistencies.
* Testability:The product of a given development phase should satisfy the conditions imposed at the start of that phase.
* Usability:It should be perfect and comfortable for users to work.
* Security:The system is completely based on the security. This system will provide security base on the password.

# CHAPTER 3

# SYTEM DESIGN

## 3.1 System Model

We consider a cloud computing architecture by combining with an example that a company uses a cloud to enable its staffs in the same group or department to share files. The system model consists of three different entities: the cloud, a group manager (i.e., the company manager), and a large number of group members (i.e., the staffs) as illustrated in Fig.



**Fig 3.1 System Model**

Cloud is operated by CSPs and provides priced abundant storage services. However, the cloud is not fully trusted by users since the CSPs are very likely to be outside of the cloud users’ trusted domain. Similarly we assume that the cloud server is honest but curious. That is, the cloud server will not maliciously delete or modify user data due to the protection of data auditing schemes, but will try to learn the content of the stored data and the identities of cloud users.

Group manager takes charge of system parameters generation, user registration, user revocation, and revealing the real identity of a dispute data owner. In the given example, the group manager is acted by the administrator of the company. Therefore, we assume that the group manager is fully trusted by the other parties.

Group members are a set of registered users that will store their private data into the cloud server and share them with others in the group. In our example, the staffs play the role of group members. Note that, the group membership is dynamically changed, due to the staff resignation and new employee participation in the company.

## 3.2 Design Goals

In this section, we describe the main design goals of the proposed scheme including access control, data confidentiality, anonymity and traceability, and efficiency as follows:

3.2.1 Access control: The requirement of access control is two fold. First, group members are able to use the cloud resource for data operations. Second, unauthorized users cannot access the cloud resource at any time, and revoked users will be incapable of using the cloud again once they are revoked.

3.2.2 Data confidentiality: Data confidentiality requires that unauthorized users including the cloud are incapable of learning the content of the stored data. An important and challenging issue for data confidentiality is to maintain its availability for dynamic groups. Specifically, new users should decrypt the data stored in the cloud before their participation, and revoked users are unable to decrypt the data moved into the cloud after the revocation.

3.3.3 Anonymity and traceability: Anonymity guarantees that group members can access the cloud without revealing the real identity. Although anonymity represents an effective protection for user identity, it also poses a potential inside attack risk to the system. For example, an inside attacker may store and share a mendacious information to derive substantial benefit. Thus, to tackle the inside attack, the group manager should have the ability to reveal the real identities of data owners.

3.3.4 Efficiency: The efficiency is defined as follows: Any group member can store and share data files with others in the group by the cloud . User revocation can be achieved without involving the remaining users. That is, the remaining users do not need to update their private keys or reencryption operations. New granted users can learn all the content data files stored before his participation without contacting with the data owner.

## 3.5 UML:

UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of the software system.

## 3.6 UML Diagrams:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

UML is a very important part of developing objects oriented software and the software development process.

UML uses mostly graphical notations to express the design of software projects.

Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

### 3.6.1 UML is a language:

It will provide vocabulary and rules for communications and function on conceptual and physical representation. So it is modeling language.

### 3.6.2 UML Specifying:

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

### 3.6.3 UML Visualization:

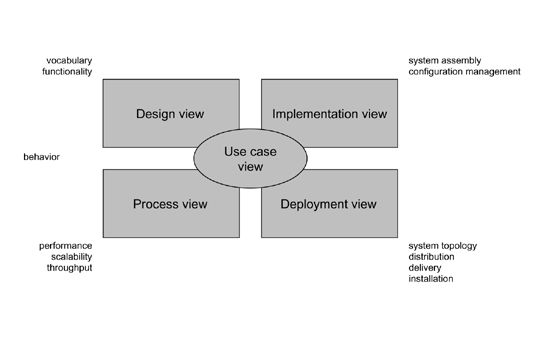
The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

### 3.6.4 UML Constructing:

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

### 3.6.5 UML Documenting:

UML provides variety of documents in addition raw executable codes.



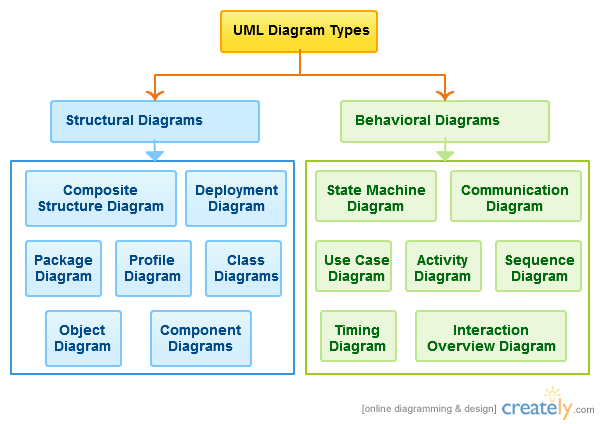
**Fig. 3.2 Modeling a System Architecture using views of UML**

The use case view of a system encompasses the use cases that describe the behavior of the system as seen by its end users, analysts, and testers.

The *design view* of a system encompasses the classes, interfaces, and collaborations that form the vocabulary of the problem and its solution.

The *process view* of a system encompasses the threads and processes that form the system's concurrency and synchronization mechanisms.

The *implementation view* of a system encompasses the components and files that are used to assemble and release the physical system.The *deployment view* of a system encompasses the nodes that form the system's hardware topology on which the system executes.



**Fig 3.3 UML diagram classification**

### 3.6.6 Uses of UML:

The UML is intended primarily for software intensive systems. It has been used as effectively for such domain

Enterprise Information System

Banking and Financial Services

Telecommunications

Transportation

Defense/Aerosp

Retails

Medical Electronics

Scientific Fields

Distributed Web

### 3.6.7 Building blocks of UML:

The vocabulary of the UML encompasses 3 kinds of building blocks

Things

Relationships

Diagrams

### Things:

Things are the data abstractions that are first class citizens in a model. Things are of 4 types

Structural Things, Behavioral Things ,Grouping Things, An notational Things

### Relationships:

Relationships tie the things together. Relationships in the UML are

Dependency, Association, Generalization, Specialization

### UML Diagrams:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).

There are two types of diagrams, they are:

Structural and Behavioral Diagrams

### Structural Diagrams:-

The UML‘s four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. ican View the static parts of a system using one of the following diagrams. Structural diagrams consists of Class Diagram, Object Diagram, Component Diagram, Deployment Diagram.

### Behavioral Diagrams :

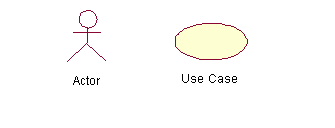
The UML’s five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML’s behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system.

Behavioral diagrams consists of

Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram

## 3.7 Use-Case diagram:

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

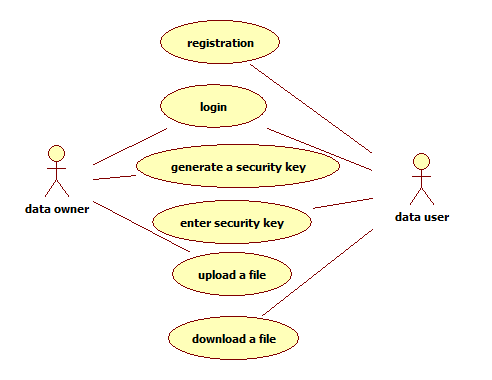
****

**Fig 3.4 Usecase diagram**

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

**Contents:**

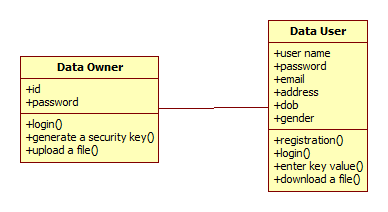
* Use cases
* Actors
* Dependency, Generalization, and association relationships
* System boundary

****

**Fig 3.5 Usecase diagram of Mona**

## 3.8 Class Diagram:

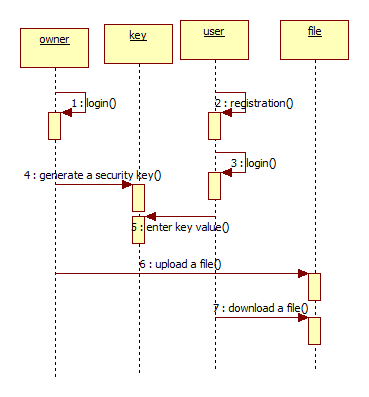
Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design. Class diagrams are arguably the most used UML diagram type. It is the main building block of any object oriented solution. It shows the classes in a system, attributes and operations of each class and the relationship between each class. In most modeling tools a class has three parts, name at the top, attributes in the middle and operations or methods at the bottom. In large systems with many classes related classes are grouped together to to create class diagrams. Different relationships between diagrams are show by different types of Arrows. Below is a image of a class diagram. Follow the link for more class diagram examples.



**Fig 3.6 Class diagram**

## 3.9 Sequence Diagram

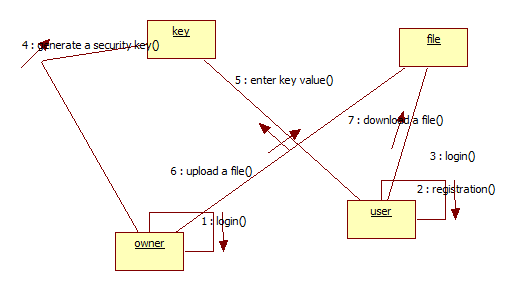
Sequence diagrams in UML shows how object interact with each other and the order those interactions occur. It’s important to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are show as arrows. This article explains thepurpose and the basics of Sequence diagrams.



**Fig 3.7 Sequence diagram**

## 3.10 Collaboration diagram

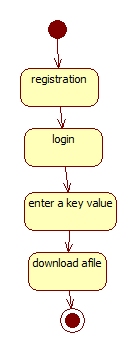
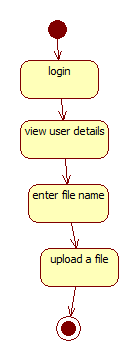
Communication diagram was called collaboration diagram in UML 1. It is similar to sequence diagrams but the focus is on messages passed between objects. The same information can be represented using a sequence diagram and different objects. Click here to understand the differences using an example.



**Fig 3.8 Collaboration diagram**

## 3.11 State machine diagrams

State machine diagrams are similar to activity diagrams although notations and usage changes a bit. They are sometime known as state diagrams or start chart diagrams as well. These are very useful to describe the behavior of objects that act different according to the state they are at the moment. Below State machine diagram show the basic states and actions.



**Fig. 3.9 State machine diagram**

## 3.12 Activity Diagram:

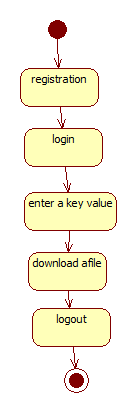
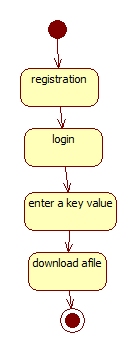
Activity diagrams describe the workflow behavior of a system.  Activity diagrams are similar to state diagrams because activities are the state of doing something.  The diagrams describe the state of activities by showing the sequence of activities performed.  Activity diagrams can show activities that are conditional or parallel.

### 3.12.1 How to Draw: Activity Diagrams

Activity diagrams show the flow of activities through the system.  Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities.  A fork is used when multiple activities are occurring at the same time.  The diagram below shows a fork after activity1.  This indicates that both activity2 and activity3 are occurring at the same time.  After activity2 there is a branch.  The branch describes what activities will take place based on a set of conditions.  All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch.   After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.   .

### 3.12.2 When to Use: Activity Diagrams

Activity diagrams should be used in conjunction with other modeling techniques such as interaction diagrams and state diagrams.  The main reason to use activity diagrams is to model the workflow behind the system being designed.  Activity Diagrams are also useful for: analyzing a use case by describing what actions need to take place and when they should occur; describing a complicated sequential algorithm; and modeling applications with parallel processes.



**Fig. 3.10 Activity diagram**

## 3.13 Component diagram

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems that has many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. Below images shows a component diagram.



**Fig. 3.11 Component diagram**

## 3.14 Deployment Diagram

A deployment diagrams shows the hardware of your system and the software in those hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration. Below is an example deployment diagram.



**Fig. 3.12 Deployment diagram**

# CHAPTER 4

# SYSTEM IMPLEMENTATION

## 4.1 Modules :

* Registration
* Login
* File Upload
* Chart Creation
* File Download
* User Deletion

### 4.1.1 Registration:

In this module an User has to register first, then only he/she has to access the data base.

### 4.1.2 Login:

In this module, any of the above mentioned person have to login, they should login by giving their email and password .

### 4.1.3 File Upload:

In this module Manager(Owner) uploads the file(along with meta data) into database, with the help of this metadata and its contents, the end user has to download the file. The uploaded file was in encrypted form, only registered user can decrypt it. Even CSP can only view the encrypted file form.

### 4.1.4 Chart Creation:

User can view the chart, which is dynamically created by calculating the size of the file.

### 4.1.5 File Download:

The Registered users can download the file and can do updates. The modified file will be uploaded into cloud server by the user.

### 4.1.6 User Deletion:

Manager (admin) can reject the user, so as that rejected user doesn’t login and access the database.

# 

# CHAPTER 5

# SCREEN SHOTS

## 5.1 Output Screens

****

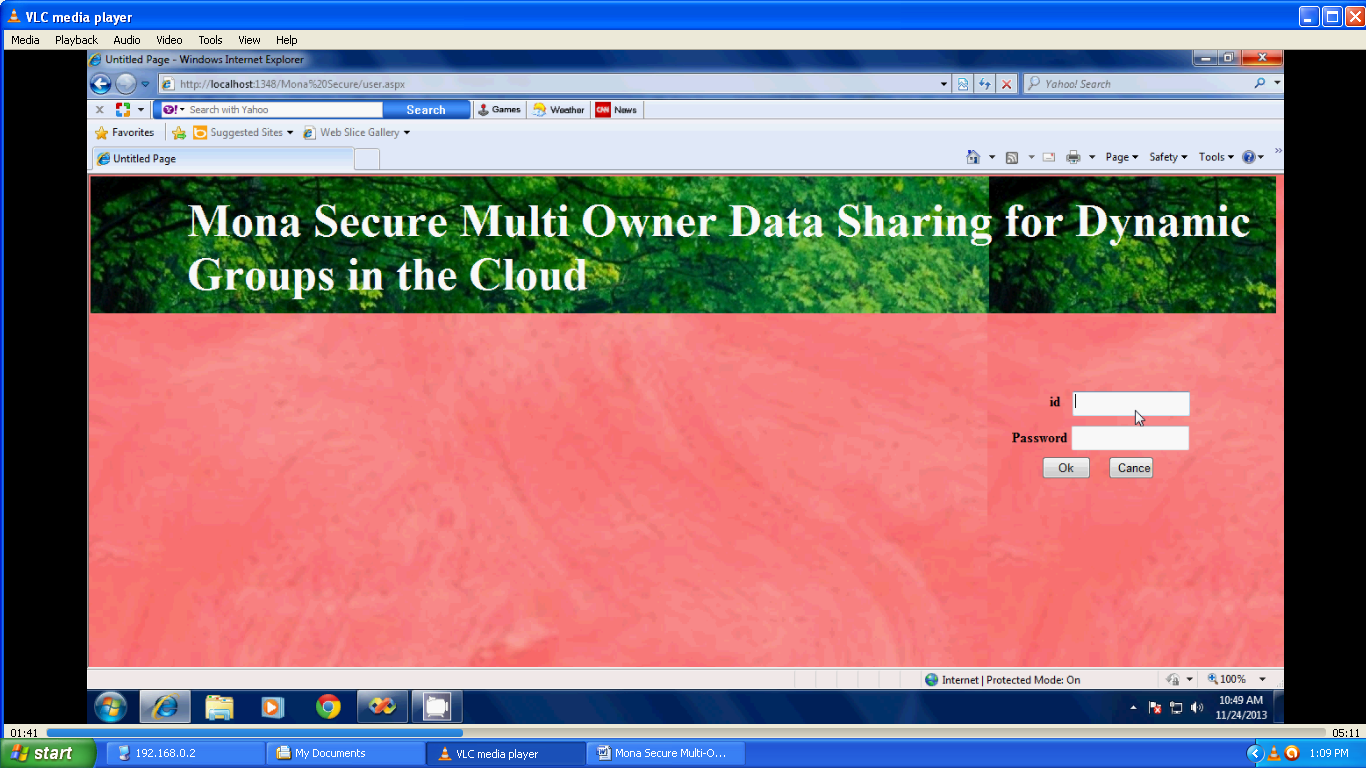
**Fig. 5.1 Home Screen**



**Fig. 5.2 Data user screen**



**Fig. 5.3 User Registration**



**Fig. 5.4 Login Screen**



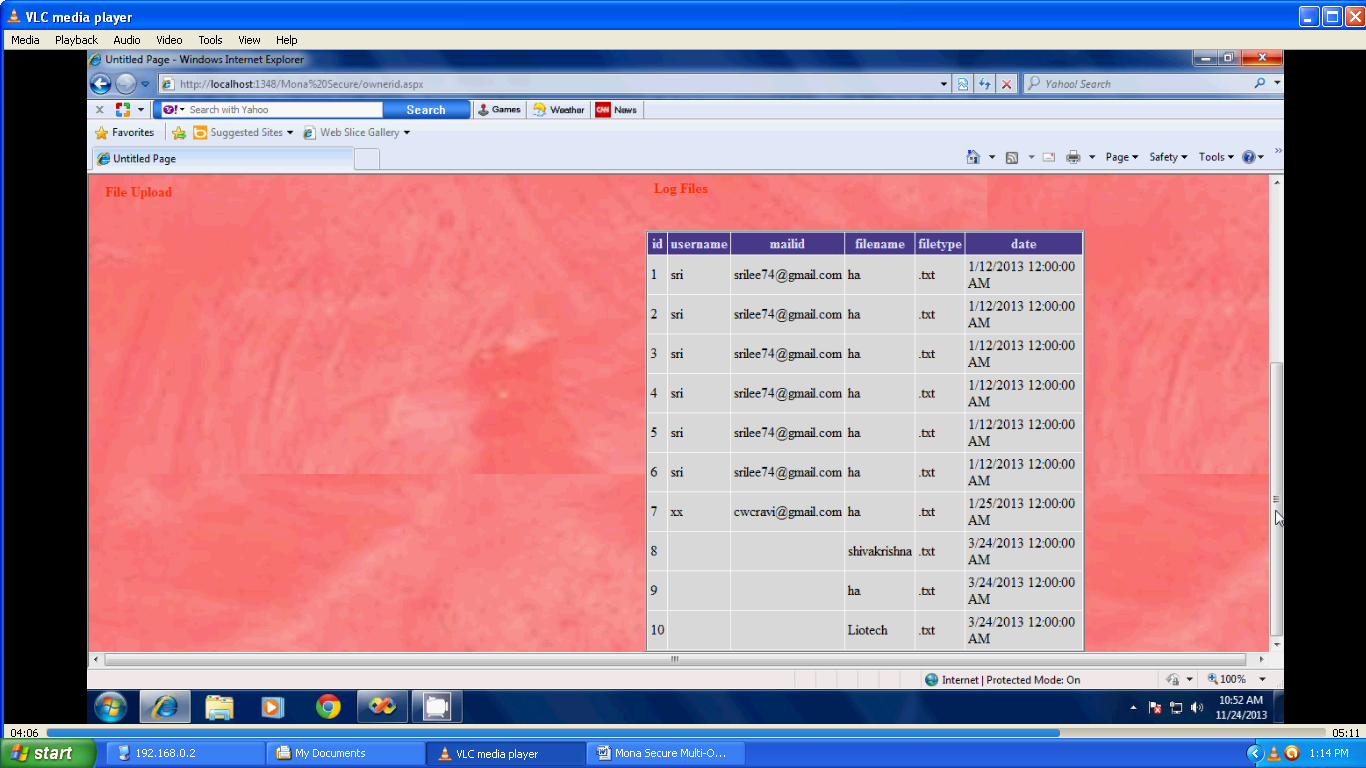
**Fig. 5.5 Login screen with username and password**



**Fig. 5.6 Enter the secret key/Signature**



**Fig. 5.7 Uploading the file**



**Fig. 5.8 Log of files uploaded**

## 5.2 Sample Code

<%--

Document : accept

Created on : Dec 1, 2013, 3:21:38 PM

Author : Thanu

--%>

<%@page import="java.sql.\*"%>

<%@pagecontentType="text/html"pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<%

String id=request.getQueryString();

String pwd=request.getParameter("Password");

//session.setAttribute("name", name);

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","secure","secure");

Statement st= con.createStatement();

int x=st.executeUpdate("update reg set status='Accept' where un='"+id+"'");

if(x==1)

{

response.sendRedirect("userkey.jsp?Message=Success");

}

else

{

response.sendRedirect("gpage.jsp?Message=Failed");

}

}

catch(Exception e)

{

e.printStackTrace();

}%>

</body>

</html>

<%--

Document : admin

Created on : Dec 1, 2013, 2:34:55 PM

Author : Thanu

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

<style>

body {

background-image: url("images/5\_1.jpg");

}

</style>

<link rel='stylesheet' type='text/css' href='styles.css' />

<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js'></script>

</head>

<body b>

<table align="center">

<tr>

<td align="center"><img src="images/3\_1.jpg" hight="500" width="900" ></td></table>

<div id='cssmenu'>

<ul>

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

<li><a href='admin.jsp'><span>Admin</span></a></li>

<li><a href='gu.jsp'><span>Group User</span></a></li>

<li class='last'><a href='gm.jsp'><span>Group Manager</span></a></li>

<li class='last'><a href='signup.jsp'><span>SignUp</span></a></li>

<li class='last'><a href='about.jsp'><span>About</span></a></li>

</ul>

</div>

<h3><center><font color="black">Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud</h3></center>

<table align="left">

<tr>

<td align="left"><img src="images/4.png" hight="200" width="200" ></td></tr>

<form action ="admincon.jsp" method="POST">

<table align="center">

<tr>

<td>User Name</td>

<td> <input type="text" name="Username" placeholder="Enter User Name" autofocus=""></td>

</tr>

<tr>

<td>Password</td>

<td><input type="password" name="Password" placeholder="Password" autofocus=""></td>

</tr>

<tr><td>

<input type="submit" value="Login">&nbsp;

<input type="reset"value="reset">

</td></tr>

</table>

</form>

</table>

</html>

</body>

</html>

<%--

Document : apage

Created on : Dec 1, 2013, 2:44:20 PM

Author : Thanu

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

<style>

body {

background-image: url("images/5\_1.jpg");

}

</style>

<link rel='stylesheet' type='text/css' href='styles.css' />

<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js'></script>

</head>

<body >

<table align="center">

<tr>

<td align="center"><img src="images/3\_1.jpg" hight="500" width="900" ></td></table>

<div id='cssmenu'>

<ul>

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

<li><a href='filedetails.jsp'><span>File Details</span></a></li>

<li><a href='userview1.jsp'><span>User Details</span></a></li>

<li class='last'><a href='index.jsp'><span>Logout</span></a></li>

</ul>

</div>

<% String name=(String)session.getAttribute("name");%>

<table align="center">

<h3><center><font color="black">Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud</h3></center>

<center> <h2>Welcome To:<%=name%></h2></center>

<tr>

<td bgcolor="lightblue"><img src="images/2.gif" hight="500" width="500" ></td></table>

</body>

</html>

<%--

Document : download

Created on : Dec 1, 2013, 5:29:13 PM

Author : Thanu

--%>

<%@page import="java.sql.\*"%>

<%@page import="databaseconnection.SendEmail" %>

<%@page import="java.sql.\*" %>

<%@page import="java.util.\*" %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<h1>Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud</h1>

<%

Statement st = null;

ResultSet rs = null;

String id=request.getQueryString();

session.setAttribute("key", id);

String email = (String)session.getAttribute("email");

out.println(email);

//String email1 = request.getParameter("select");

if(email!=""){

try{

Random rand = new Random();

int num = rand.hashCode();

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","secure","secure");

String s = "insert into random1(email,rkey) values('"+email+"',"+num+")";

PreparedStatement ps = con.prepareStatement(s);

ps.executeUpdate();

SendEmail e=new SendEmail();

e.Email(email,num);

out.println(email);

response.sendRedirect("download1.jsp?Message=Your Secretkey Successfully Sent to MailId ");

}catch(Exception e){

out.println(e);

}

}else{

out.println("ENTER USERNAME AND EMAIL");

Thread.sleep(1000);

response.sendRedirect("user.jsp?Message=Failed");

}

%>

</body>

</html>

<%--

Document : download1

Created on : Dec 1, 2013, 5:47:43 PM

Author : Thanu

--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

<style>

body {

background-image: url("images/5\_1.jpg");

}

</style>

<link rel='stylesheet' type='text/css' href='styles.css' />

<script src='http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js'></script>

</head>

<body >

<table align="center">

<tr>

<td align="center"><img src="images/3\_1.jpg" hight="500" width="900" ></td></table>

<div id='cssmenu'>

<ul>

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

<li><a href='upload.jsp'><span>File Upload</span></a></li>

<li><a href='viewfiles.jsp'><span>View Group Files</span></a></li>

<li><a href='update.jsp'><span>Update Profile</span></a></li>

<li class='last'><a href='index.jsp'><span>Logout</span></a></li>

</ul>

</div>

<h3><center><font color="black">Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud</h3></center>

<table align="left">

<tr>

<td align="left"><img src="images/4.png" hight="200" width="200" ></td></tr>

<form action="download2.jsp" method="post">

<table align="center">

<tr>

<td><font face="Times New Roman" size="+1"><strong>Enter Secret Key</strong></font></td>

<td>&nbsp;&nbsp;&nbsp;&nbsp; <input type="text" name="email" class="b"></td>

</tr>

<tr>

<td align="center"> <input type="Reset" value="Reset" class="b"></td>&nbsp;&nbsp;

<td><input type="submit" value="Login" class="b">

</tr>

</table>

</form><br><br><br>

</table>

</body>

</html>

# 

# CHAPTER 6

# TESTING

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

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

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

## 6.1 Types of tests

### 6.1.1 Unit testing

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

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

### 6.1.2 Integration testing

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

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error

### 6.1.3 Acceptance testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

### 6.1.4 Functional test

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

### 6.1.5 System Test

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

### 6.1.6 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

### 6.1.7 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

## 6.2 Testing strategy

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

## 6.3 Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

## 6.4 Features to be tested

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

## 6.5 Test Results:

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

**Table 6.1 Testing Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | Test case | Input | Expected output | Obtained Output |
| 1. | Owner Login | Owner Id and password | Owner page is open | Owner page is open |
| 2. | User Login | User id and password | User page is open | User page is  open |
| 3. | File | Select a file | Enter the file name | Upload the file |
| 4. | Key | Owner generate a key | Send to a mail | View the key  value |
| 5. | Upload | Search a file | Select the file | Upload a file |
| 6. | Download | Select a file | Select file name | Download a file |

# 

# CONCLUSION

In this paper, we design a secure data sharing scheme, Mona, for dynamic groups in an untrusted cloud. In Mona, a user is able to share data with others in the group without revealing identity privacy to the cloud. Additionally, Mona supports efficient user revocation and new user joining. More specially, efficient user revocation can be achieved through a public revocation list without updating the private keys of the remaining users, and new users can directly decrypt files stored in the cloud before their participation. Moreover, the storage overhead and the encryption computation cost are constant. Extensive analyses show that our proposed scheme satisfies the desired security requirements and guarantees efficiency as well.

# REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A View of Cloud Computing,” Comm. ACM, vol. 53, no. 4, pp. 50-58, Apr. 2010.

[2] S. Kamara and K. Lauter, “Cryptographic Cloud Storage,” Proc. Int’l Conf. Financial Cryptography and Data Security (FC), pp. 136- 149, Jan. 2010.

[3] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing,” Proc. IEEE INFOCOM, pp. 534-542, 2010.

[4] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, “Plutus: Scalable Secure File Sharing on Untrusted Storage,” Proc. USENIX Conf. File and Storage Technologies, pp. 29-42, 2003.

[5] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, “Sirius: Securing Remote Untrusted Storage,” Proc. Network and Distributed Systems Security Symp. (NDSS), pp. 131-145, 2003.

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage,” Proc. Network and Distributed Systems Security Symp. (NDSS), pp. 29-43, 2005.

[7] R. Lu, X. Lin, X. Liang, and X. Shen, “Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing,” Proc. ACM Symp. Information, Computer and Comm.

Security, pp. 282-292, 2010.

[8] B. Waters, “Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization,” Proc. Int’l Conf. Practice and Theory in Public Key Cryptography Conf. Public Key Cryptography, http://eprint.iacr.org/2008/290.pdf, 2008.

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data,” Proc. ACM Conf. Computer and Comm. Security (CCS),

pp. 89-98, 2006. ess Receivers,” Proc. Ann. Int’l Cryptology Conf. Advances in Cryptology (CRYPTO), pp. 41-62, 2001.