KASA

NANDAKUMAR. S. K

School of CS and IT, Jain Deemed-to-be University, Bangalore, Karnataka, India

Dr.BHUVANA

School of Computer Science and IT, Jain Deemed-to-be University, Bangalore, Karnataka, INDIA

Abstract

The capability of selectively sharing encrypted data with different users via public cloud storage may greatly ease security concerns over inadvertent data leaks in the cloud. A key challenge to designing such encryption schemes lies in the efficient management of encryption keys. The desired flexibility of sharing any group of selected documents with any group of users demands different encryption keys to be used for different documents. However, this also implies the necessity of securely distributing to users a large number of keys for both encryption and search, and those users will have to securely store the received keys, and submit an equally large number of keyword trapdoors to the cloud in order to perform search over the shared data. The implied need for secure communication, storage, and complexity clearly renders the approach impractical. In this paper, we address this practical problem, which is largely neglected in the literature, by proposing the novel concept of *key aggregate searchable encryption (KASE)* and instantiating the concept through a concrete KASE scheme, in which a data owner only needs to distribute a single key to a user for sharing a large number of documents, and the user only needs to submit a single trapdoor to the cloud for querying the shared documents. The security analysis and performance evaluation both confirm that our proposed schemes are provably secure and practice ally efficient.

Introduction

Cloud storage has emerged as a promising solution for providing ubiquitous, convenient, and on-demand accesses to large amounts of data shared over the Internet. Today, millions of users are sharing personal data, such as photos and videos, with their friends through social network applications based on cloud storage on a daily basis. Business users are also being attracted by cloud storage due to its numerous benefits, including lower cost, greater agility, and better resource utilization.

Problem Statement

Convenience of sharing data via cloud storage, users is also increasingly concerned about inadvertent data leaks in the cloud. Such data leaks, caused by a malicious adversary or a misbehaving cloud operator, can usually lead to serious breaches of personal privacy or business secrets to address users' concerns over potential data leaks in cloud storage.

System Analysis

Introduction to System Analysis

System

A system is an orderly group of interdependent components linked together according to a plan to achieve a specific objective. Its main characteristics are organization, interaction, interdependence, integration and a central objective.

System Analysis

System analysis and design are the application of the system approach to problem solving generally using computers. To reconstruct a system the analyst must consider its elements output and inputs, processors, controls feedback and environment.

Analysis

Analysis is a detailed study of the various operations performed by a system and their relationships within and outside of the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate system should consider other related systems. During analysis data are collected on the available files decision points and transactions handled by the present system. This involves gathering information and using structured tools for analysis.

Existing System

Convenience of sharing data via cloud storage, users is also increasingly concerned about inadvertent data leaks in the cloud. Such data leaks, caused by a malicious adversary or a misbehaving cloud operator, can usually lead to serious breaches of personal privacy or business secrets To address users' concerns over potential data leaks in cloud storage , a common approach is for the data owner to encrypt all the data before uploading them to the cloud, such that later the encrypted data may be retrieved and decrypted by those who have the decryption keys. Such cloud storage is often called the cryptographic cold storage. However, the encryption of data makes it challenging for users to search and then selectively retrieve only the data containing given keywords. A common solution is to employ a searchable encryption (SE) scheme in which the data owner is required to encrypt potential keywords and upload them to the cloud together with encrypted data, such that, for retrieving data matching a keyword, the user will send the corresponding keyword *trapdoor* to the cloud for performing search over the encrypted data. Although combining a searchable encryption scheme with cryptographic cloud storage can achieve the basic security requirements of a cloud storage, implementing such a system for large scale applications involving millions of users and billions of files may still be hindered by practical issues involving the efficient management of encryption keys, which to the best of our knowledge, are largely ignored in the literature. First of all, the need for selectively sharing encrypted data with different usually demands different encryption keys to be used for different files. However, this implies the number of keys that need to be distributed to users, both for them to search over the encrypted files and to decrypt the files, will be proportional to the number of such files. Such a large number of keys must not only be distributed to users via secure channels, but also be securely stored and managed by the users in their devices. In addition, a large number of trapdoors must be generated by users and submitted to the cloud in order to perform a keyword search over many files. The implied need for secure communication, storage, and computational complexity may render such a system inefficient and impractical.

Disadvantages of the Existing System

- Large number of trapdoors must be generated by users and submitted to the cloud in order to perform keyword search over many files.
- This implied need for secure communication, storage and computational complexity may render such a system inefficient and impractical

Proposed System

In this paper, we address this challenge by proposing the novel concept of key-aggregate searchable encryption and instantiating the concept through a concrete KASE scheme. The proposed KASE scheme applies to any cloud storage that supports the searchable group data sharing functionality, which means any user may selectively share a group of selected files with a group of selected users, while allowing the latter to perform keyword search over the former. To support searchable group data sharing the main requirements for efficient key management are twofold. First, a data owner only needs to distribute a single aggregate key to a user for sharing any number of files. Second, the user only needs to submit a single aggregate trapdoor to the cloud for performing keyword search over any number of shared files. To the best of our knowledge, the KASE scheme proposed in this paper is the first known scheme that can satisfy both requirements

Advantages of the Proposed System

- Both functional and security requirements for designing a valid KASE scheme.
- Efficiency
- Security

METHODOLOGY USED:

- ➢ Key aggregate searchable encryption (KASE)
- ➢ Keyword trapdoor
- ➢ Real Cloud Storage

FEASIBILITY STUDY

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility Study. This type of study if a project can and should be taken. In the conduct of the feasibility study, the analyst will usually consider seven distinct, but inter-related types of feasibility.

Technical Feasibility

This is considered with specifying equipment and software that will successful satisfy the user requirement the technical needs of the system may vary considerably but might include

- The facility to produce outputs in a given time.
- Response time under certain conditions.
- Ability to process a certain column of transaction at a particular speed.

Economic Feasibility

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. More commonly known as cost / benefit analysis. The procedure is to determine the benefits and savings are expected form a proposed system and a compare them with costs. It benefits outweigh costs; a decision is taken to design and implement the system will have to be made if it is to have a chance of being approved. There is an ongoing effort that improves in accuracy at each phase of the system life cycle.

Operational Feasibility

It is mainly related to human organization and political aspects. These points are considered are

- *What changes will be brought with the system?*

- *What organizational structures are distributed?*

- *What new skills will be required?*

- *Do the existing system staff members have these skills?*

- *If not, can they be trained in the course of time?*

System Design

6.1 Logical Design

Design for WebApps encompasses technical and non-technical activities. The look and feel of content is developed as part of graphic design; the aesthetic layout of the user interface is created as part of interface design; and the technical structure of the WebApp is modeled as part of architectural and navigational design.

This argues that a Web engineer must design an interface so that it answers three primary questions for the end-user:

1. *Where am I?* – The interface should (1) provide an indication of the WebApp has been accessed and (2) inform the user of her location in the content.

2. *What can I do now?* – The interface should always help the user understand his current options- what functions are available, what links are live, what content is relevant.

3. *Where have I been; where am I going?* – The interface must facilitate navigation. Hence it must provide a “map” of where the user has been and what paths may be taken to move elsewhere in the WebApp.

6.2 Design goals – the following are the design goals that are applicable to virtually every WebApp regardless of application domain, size, or complexity.

1. Simplicity
2. Consistency
3. Identity
4. Visual appeal
5. Compatibility.

Design leads to a model that contains the appropriate mix of aesthetics, content, and technology. The mix will vary depending upon the nature of the WebApp, and as a consequence the design activities that are emphasized will also vary.

The activities of the Design process:

1. Interface design-describes the structure and organization of the user interface. Includes a representation of screen layout, a definition of the modes of interaction, and a description of navigation mechanisms. Interface Control mechanisms- to implement navigation options, the designer selects form one of a number of interaction mechanism;

   a. Navigation menus
   b. Graphic icons
   c. Graphic images

   Interface Design work flow- the work flow begins with the identification of user, task, and environmental requirements. Once user tasks have been identified, user scenarios are created and analyzed to define a set of interface objects and actions.

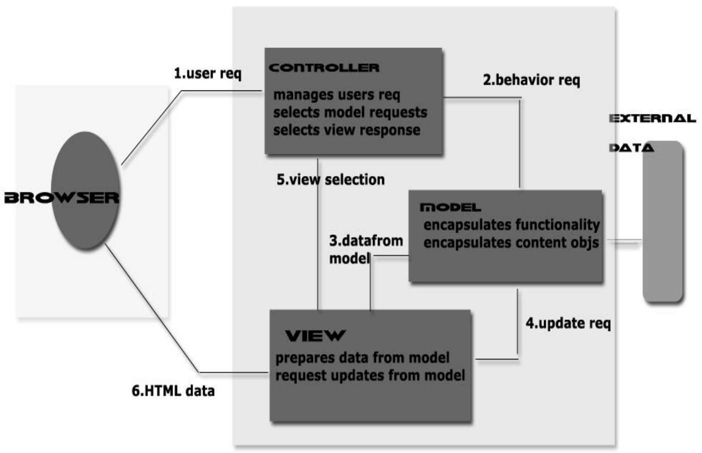
2. Aesthetic design-also called graphic design, describes the "look and feel" of the WebApp. Includes color schemes, geometric layout. Text size, font and placement, the use of graphics, and related aesthetic decisions.

3. Content design-defines the layout, structure, and outline for all content that is presented as part of the WebApp. Establishes the relationships between content objects.

4. Navigation design-represents the navigational flow between contents objects and for all WebApp functions.

5. Architecture design-identifies the overall hypermedia structure for the WebApp. Architecture design is tied to the goals establish for a WebApp, the content to be presented, the users who will visit, and the navigation philosophy that has been established.

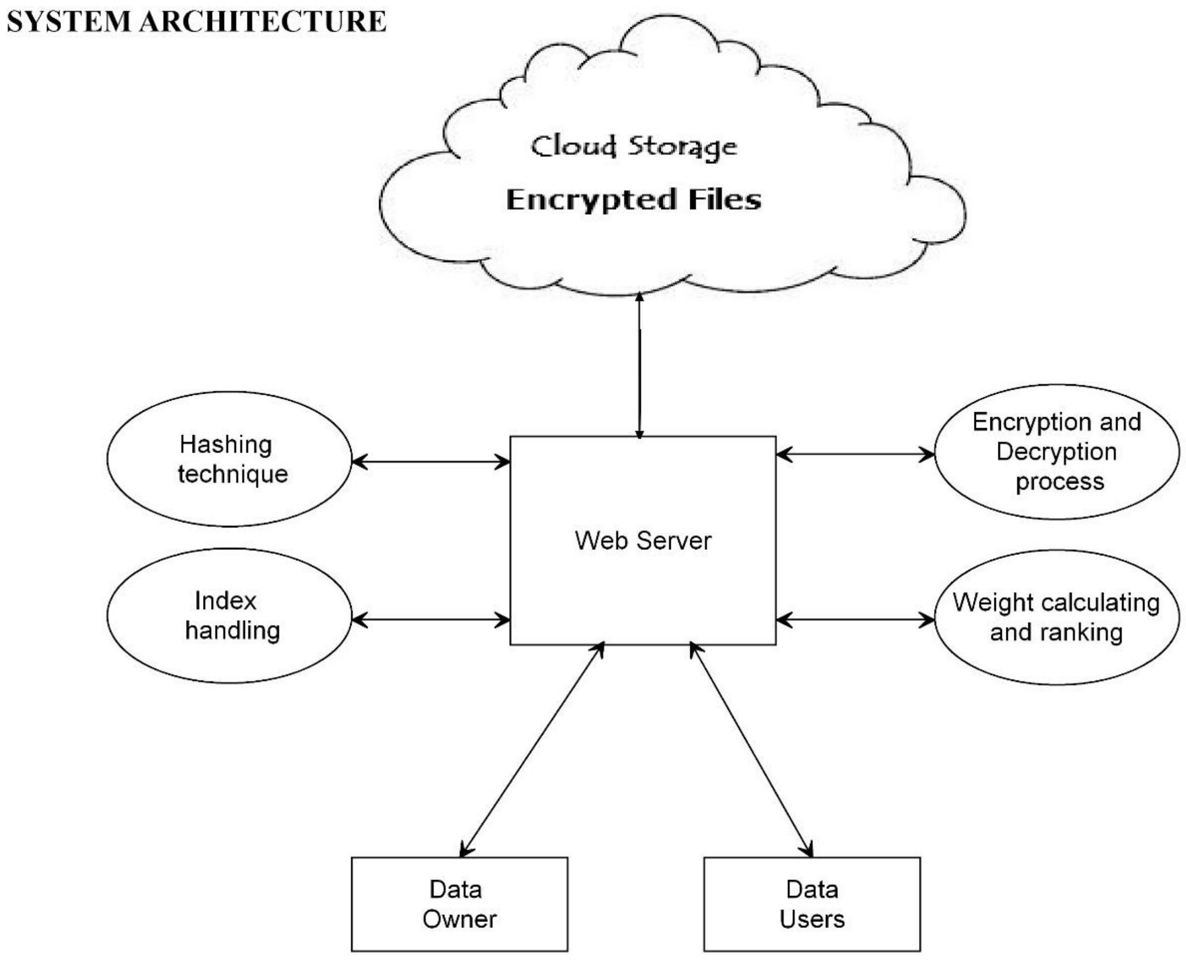
   a. Content architecture, focuses on the manner in which content objects and structured for presentation and navigation.
   b. WebApp architecture, addresses the manner in which the application is structure to manage user interaction, handle internal processing tasks, effect navigation, and present content. WebApp architecture is defined within the context of the development environment in which the application is to be implemented.



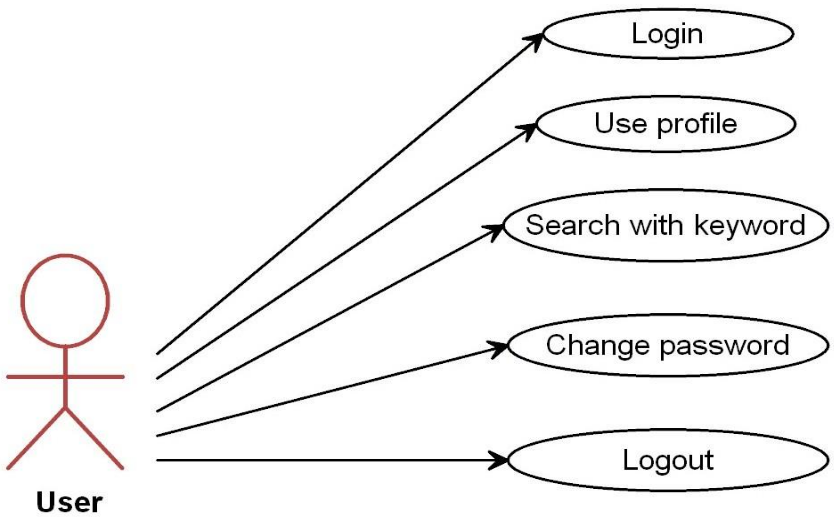
J2EE uses MVC Architecture

6. Component design-develops the detailed processing logic required to implement functional components.

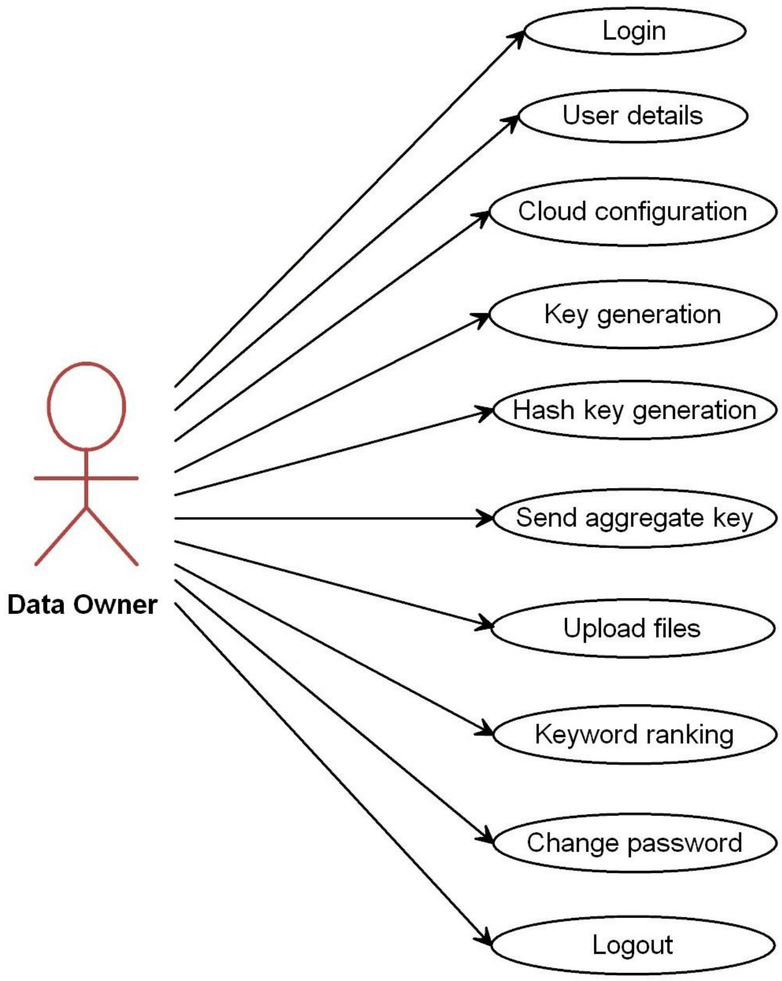
System Architecture



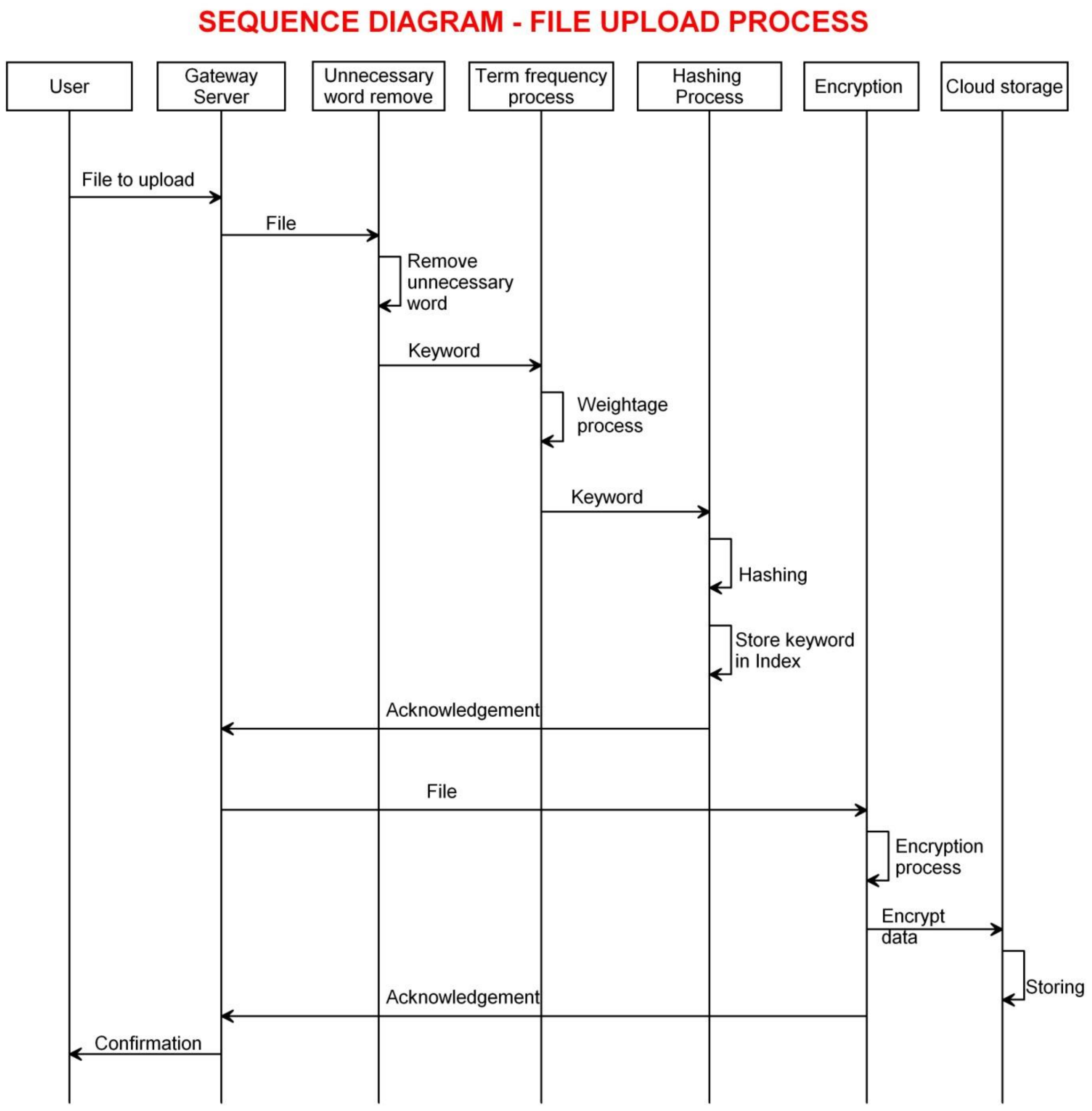
USE CASE DIAGRAM



USE CASE DIAGRAM



Sequence Diagram



Testing

Definition

Unit testing is a development procedure where programmers create tests as they develop software. The tests are simple short tests that test functionally of a particular unit or module of their code, such as a class or function.

Using open source libraries like cunit, oppunit and nun it (for C, C++ and C#) these tests can be automatically run and any problems found quickly. As the tests are developed in parallel with the source unit test demonstrates its correctness.

Validation and System Testing

Validation testing is a concern which overlaps with integration testing. Ensuring that the application fulfils its specification is a major criterion for the construction of an integration test. Validation testing also overlaps to a large extent with ***System Testing***, where the application is tested with respect to its typical working environment. Consequently for many processes no clear division between validation and system testing can be made. Specific tests which can be performed in either or both stages include the following.

- **Regression Testing:** Where this version of the software is tested with the automated test harness used with previous versions to ensure that the required features of the previous version are skill working in the new version.

- **Recovery Testing:** Where the software is deliberately interrupted in a number of ways off, to ensure that the appropriate techniques for restoring any lost data will function.

- **Security Testing:** Where unauthorized attempts to operate the software, or parts of it, attempted it might also include attempts to obtain access the data, or harm the software installation or even the system software. As with all types of security determined will be able to obtain unauthorized access and the best that can be achieved is to make this process as difficult as possible.

- **Stress Testing:** Where abnormal demands are made upon the software by increasing the rate at which it is asked to accept, or the rate t which it is asked to produce information. More complex tests may attempt to crate very large data sets or cause the soft wares to make excessive demands on the operating system.

- **Performance testing:** Where the performance requirements, if any, are checked. These may include the size of the software when installed, type amount of main memory and/or secondary storage it requires and the demands made of the operating when running with normal limits or the response time.

- **Usability Testing:** The process of usability measurement was introduced in the previous chapter. Even if usability prototypes have been tested whilst the application was constructed, a validation test of the finished product will always be required.

- **Alpha and beta testing:** This is where the software is released to the actual end users. An initial release, the alpha release, might be made to selected users who be expected to report bugs and other detailed observations back to the production team. Once the application changes necessitated by the alpha phase can be made to larger more representative set users, before the final release is made to all users.

  The final process should be a **Software audit** where the complete software project is checked to ensure that it meets production management requirements. This ensures that all required documentation has been produced, is in the correct format and is of acceptable quality. The purpose of this review is: firstly to assure the quality of the production process and by implication construction phase commences. A formal hand over from the development team at the end of the audit will mark the transition between the two phases.

- **Integration Testing**:

Integration Testing can proceed in a number of different ways, which can be broadly characterized as top down or bottom up. In top down integration testing the high level control routines are tested first, possibly with the middle level control structures present only as stubs. Subprogram stubs were presented in section2 as incomplete subprograms which are only present to allow the higher. Level control routines to be tested.

Top down testing can proceed in a **depth-first** or a **breadth-first** manner. For depth-first integration each module is tested in increasing detail, replacing more and more levels of detail with actual code rather than stubs. Alternatively breadth-first would processed by refining all the modules at the same level of control throughout the application .in practice a combination of the two techniques would be used. At the initial stages all the modules might be only partly functional, possibly being implemented only to deal with non-erroneous data. These would be tested in breadth-first manner, but over a period of time each would be replaced with successive refinements which were closer to the full functionality. This allows depth-first testing of a module to be performed simultaneously with breadth-first testing of all the modules.

The other major category of integration testing is ***Bottom Up Integration Testing*** where an individual module is tested form a test harness. Once a set of individual module have been tested they are then combined into a collection of modules ,known as builds, which are then tested by a second test harness. This process can continue until the build consists of the entire application. In practice a combination of top down and bottom-up testing would be used. In a large software project being developed by a number of sub-teams, or a smaller project where different modules were built by individuals. The sub teams or individuals would conduct bottom-up testing of the modules which they were constructing before releasing them to an integration team which would assemble them together for top-down testing.

- **Unit Testing:**

Unit testing deals with testing a unit as a whole. This would test the interaction of many functions but confine the test within one unit. The exact scope of a unit is left to interpretation. Supporting test code, sometimes called ***Scaffolding***, may be necessary to support an individual test. This type of testing is driven by the architecture and implementation teams. This focus is also called black-box testing because only the details of the interface are visible to the test. Limits that are global to a unit are tested here.

In the construction industry, scaffolding is a temporary, easy to assemble and disassemble, frame placed around a building to facilitate the construction of the building. The construction workers first build the scaffolding and then the building. Later the scaffolding is removed, exposing the completed building.similarly, in software testing, one particular test may need some supporting software. This software establishes can a correct evaluation of the test take place. The scaffolding software may establish state and values for data structures as well as providing dummy external functions for the test. Different scaffolding software may be needed form one test to another test. Scaffolding software rarely is considered part of the system.

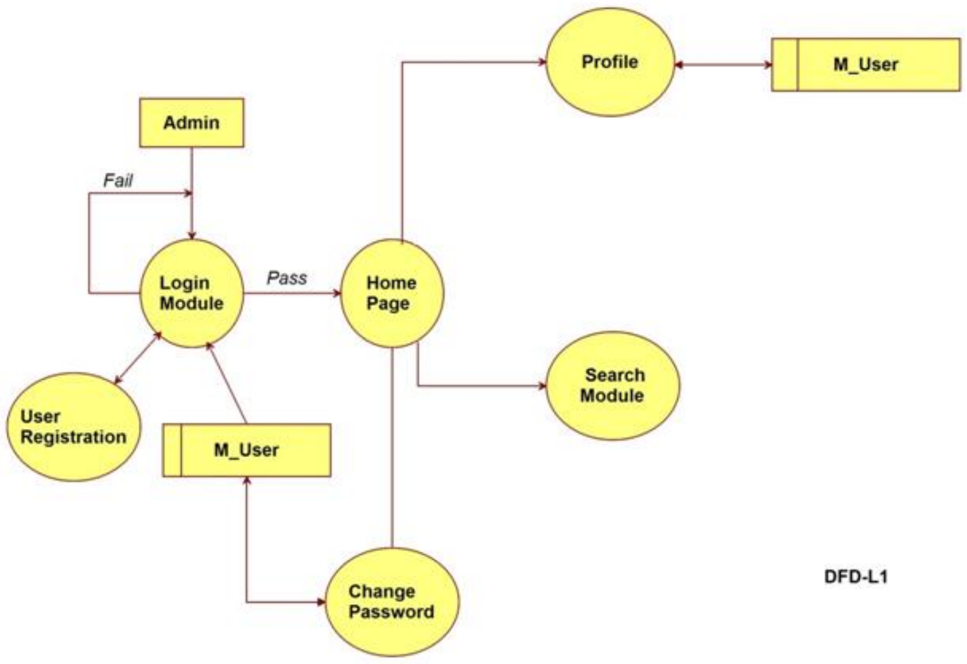
Some times the scaffolding software becomes larger than the system software being tested. Usually the scaffolding software is not of the same quality as the system software and frequently is quite fragile. A small change in test may lead to much larger changes in the scaffolding.

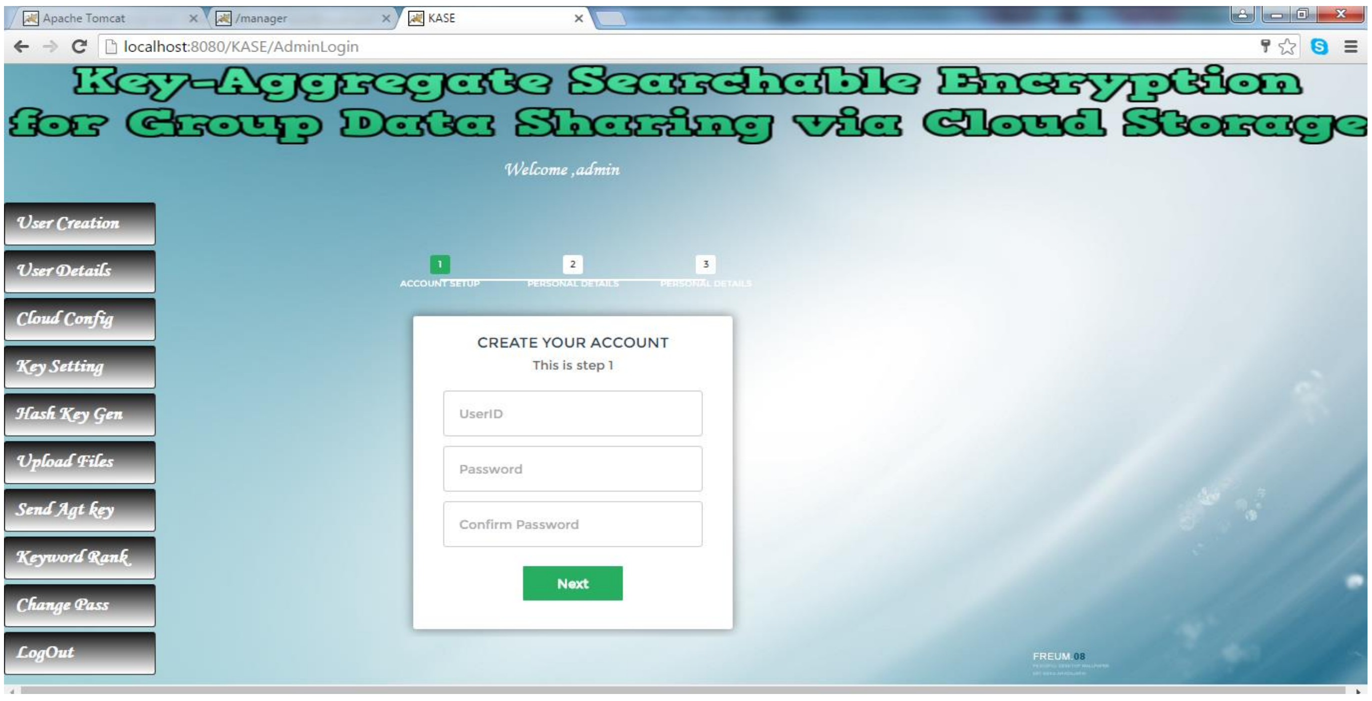
Internal and unit testing can be automated with the help of coverage tools. Analyzes the source code and generated a test that will execute every alternative thread of execution. Typically, the coverage tool is used in a slightly different way. First the coverage tool is used to augment the source by placing information prints after each line of code. Then the testing suite is executed generating an audit trail. This audit trail is analyzed and reports the percent of the total system code executed during the test suite. If the coverage is high and the untested source lines are of low impact to the system's overall quality, then no more additional tests are required.

Conclusion

Considering the practical problem of privacy preserving data sharing system based on public cloud storage which requires a data owner to distribute a large number of keys to users to enable them to access his/her documents, we for the first time propose the concept of key-aggregate searchable encryption and construct a concrete KASE scheme. Both analysis and evaluation results confirm that our work can provide an effective solution to building practical data sharing system based on public cloud storage.

DFD-DATA USER SESSION



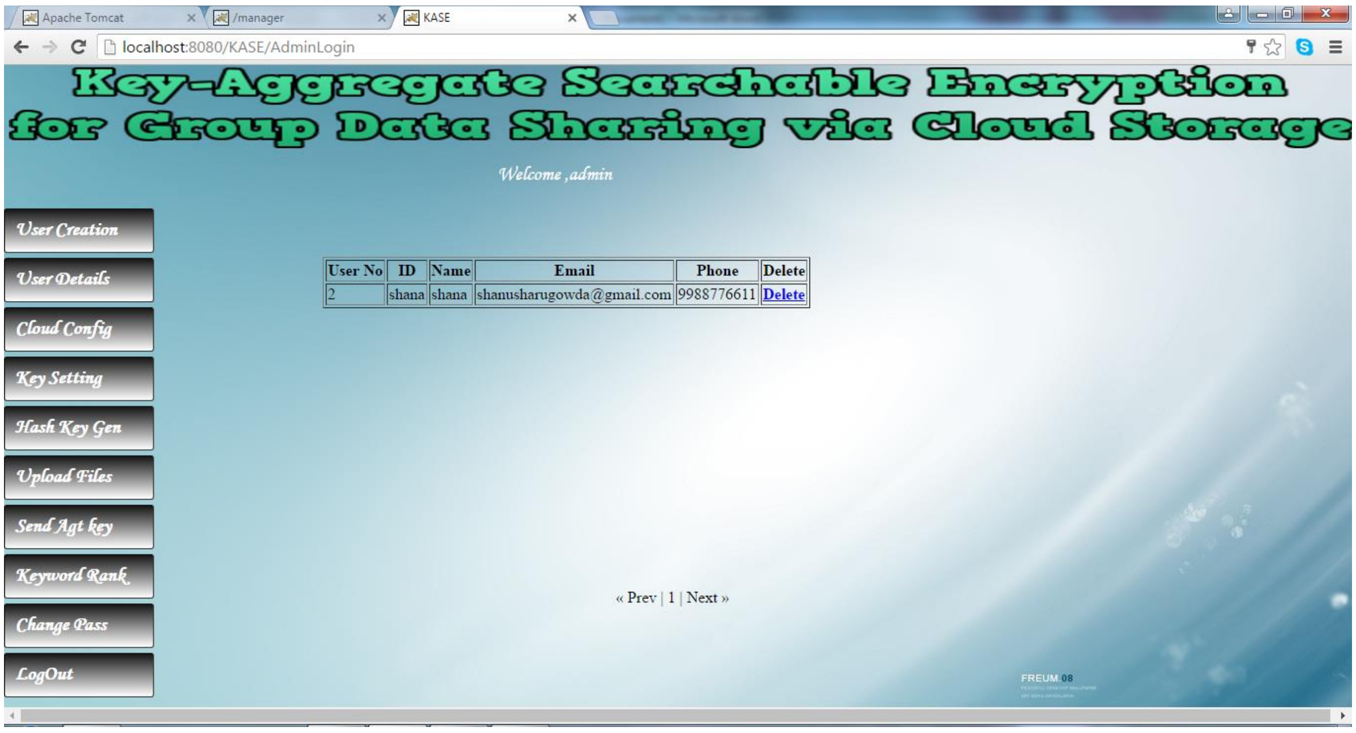


Key-Aggregate Searchable Encryption for Group Data Sharing via Cloud Storage

Welcome ,admin

CREATE YOUR ACCOUNT

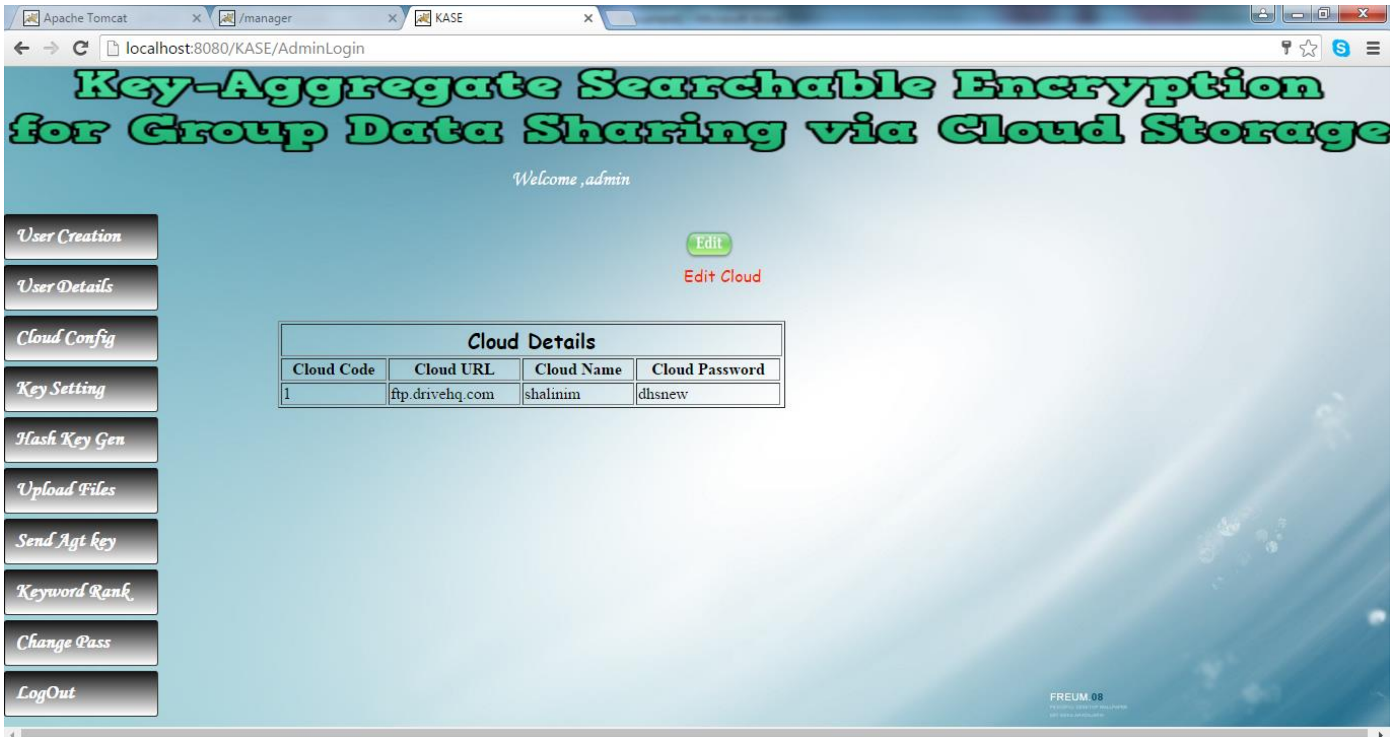
This is step 1



Key-Aggregate Searchable Encryption for Group Data Sharing via Cloud Storage

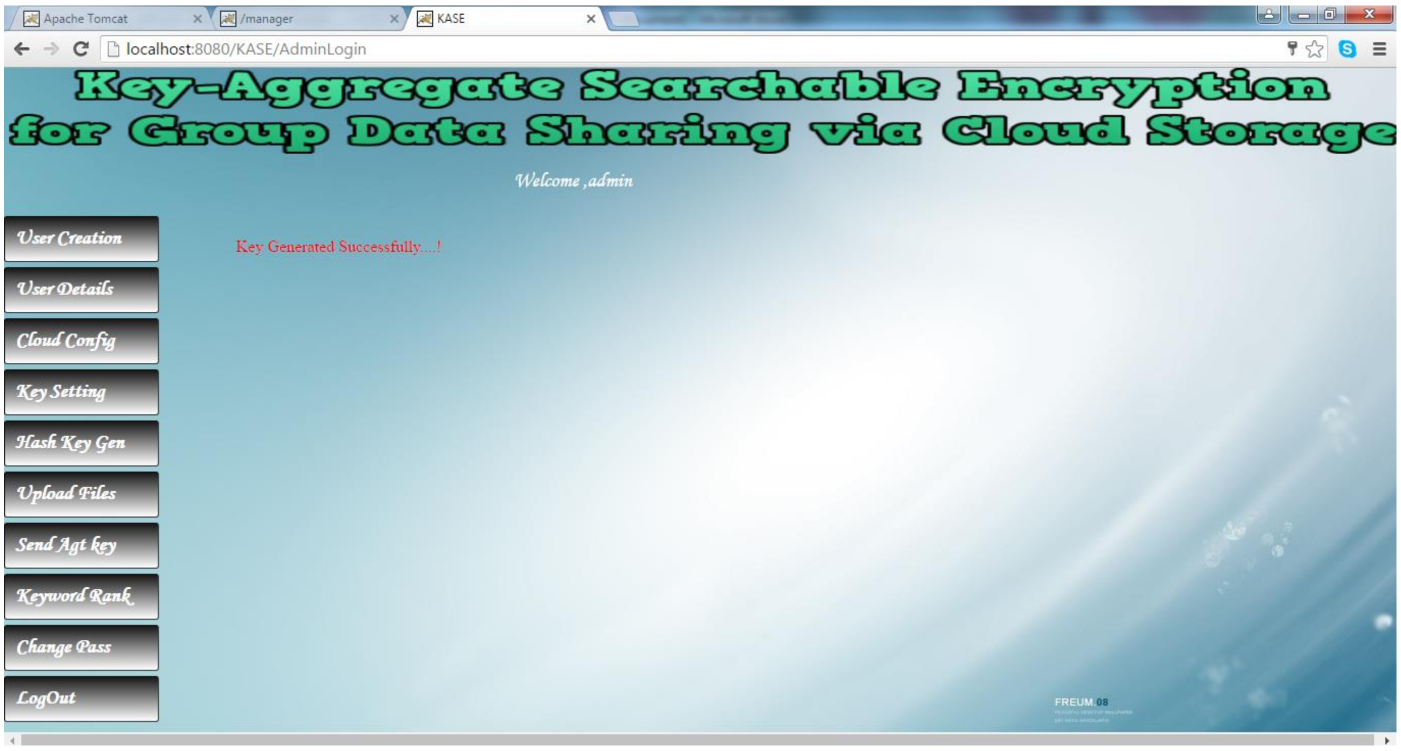
Welcome ,admin

User No	ID	Name	Email	Phone	Delete
2	shana	shana	shanusharugowda@gmail.com	9988776611	Delete

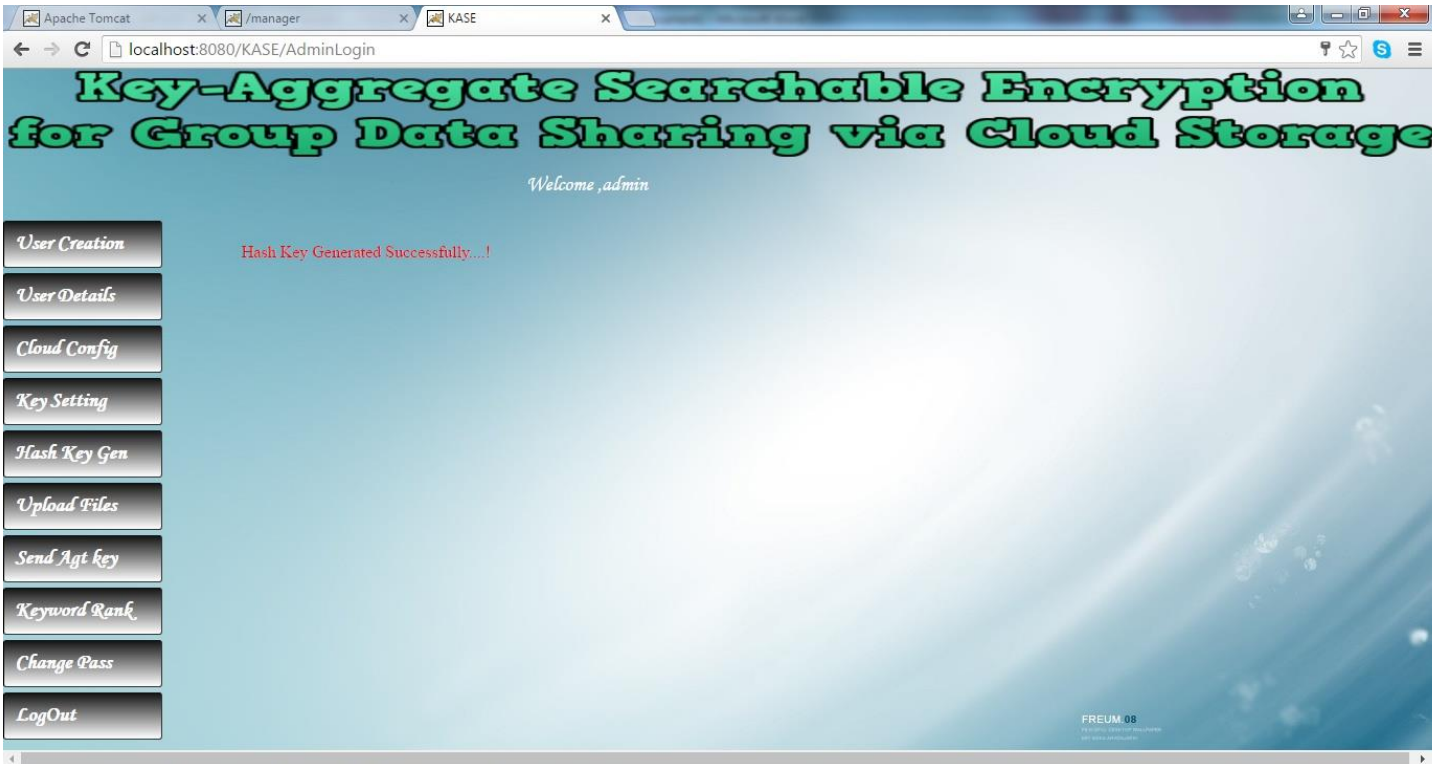


Cloud Details

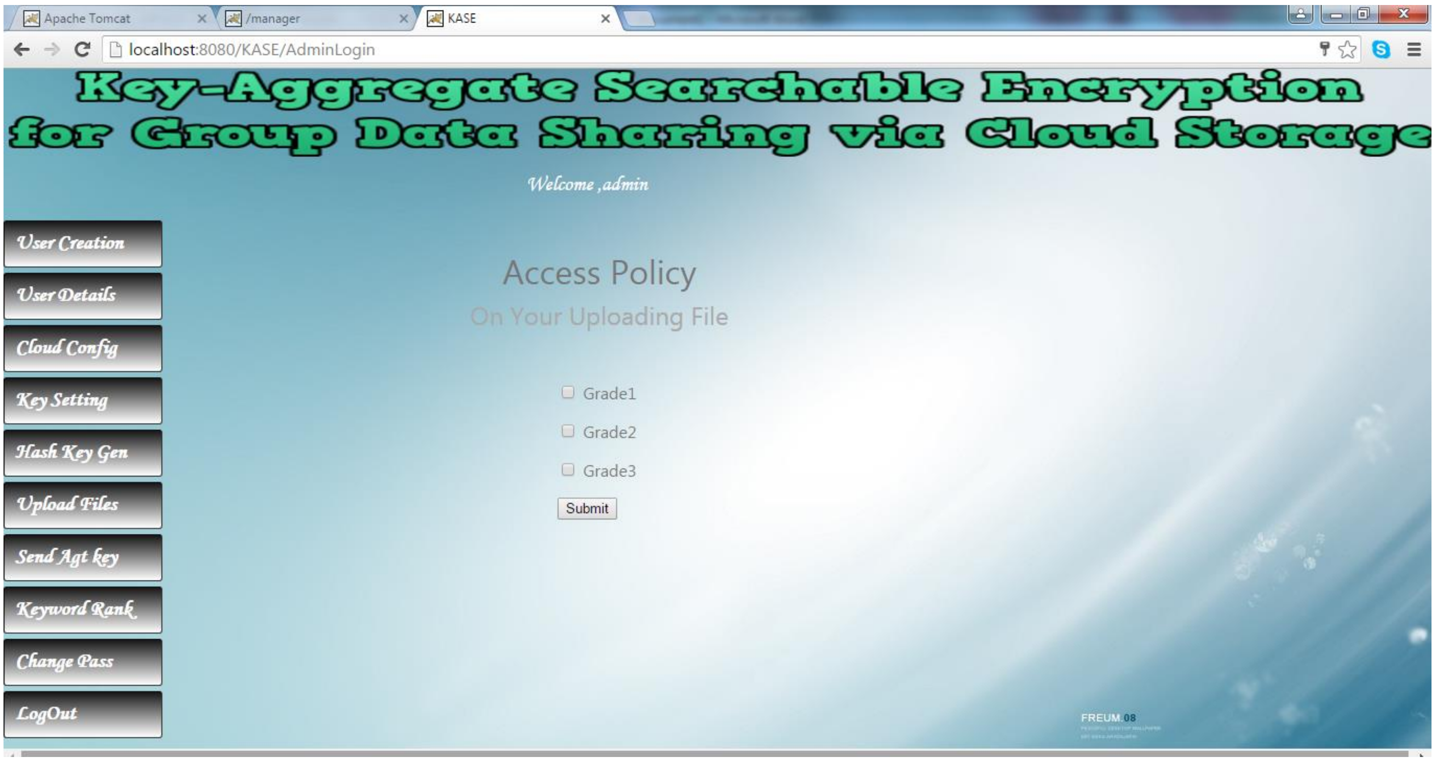
Cloud Code	Cloud URL	Cloud Name	Cloud Password
1	ftp.drivehq.com	shalinim	dhsnew



Key Generated Successfully....!

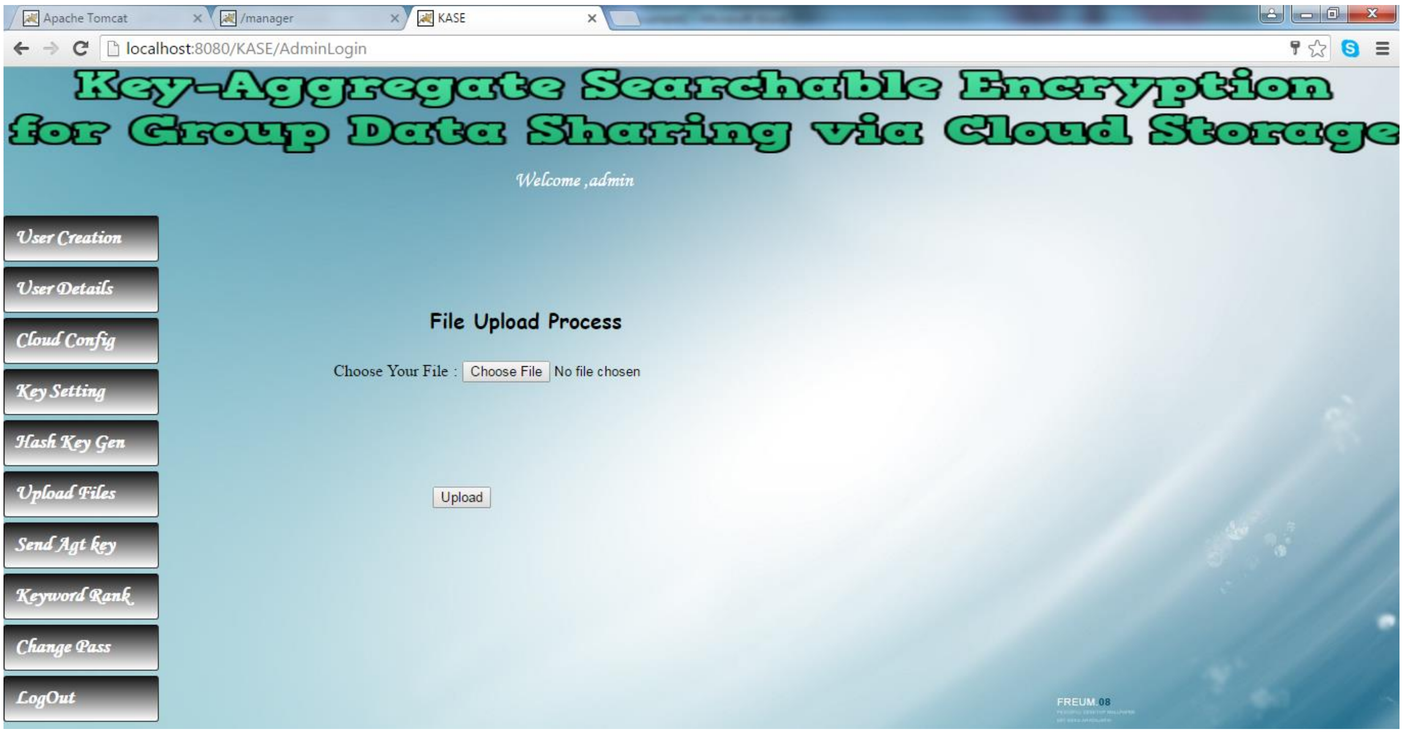


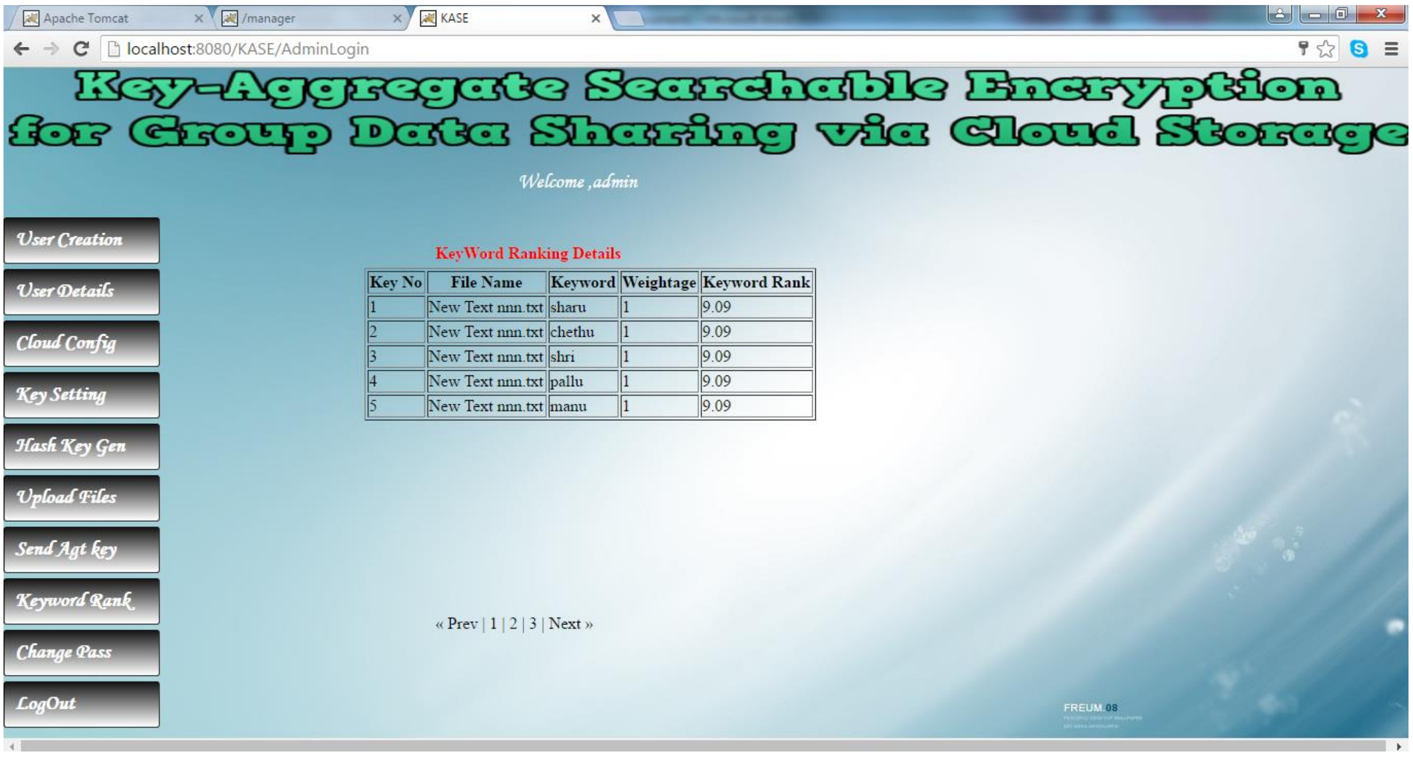
Hash Key Generated Successfully....!



Access Policy
On Your Uploading File

- [ ] Grade1
- [ ] Grade2
- [ ] Grade3

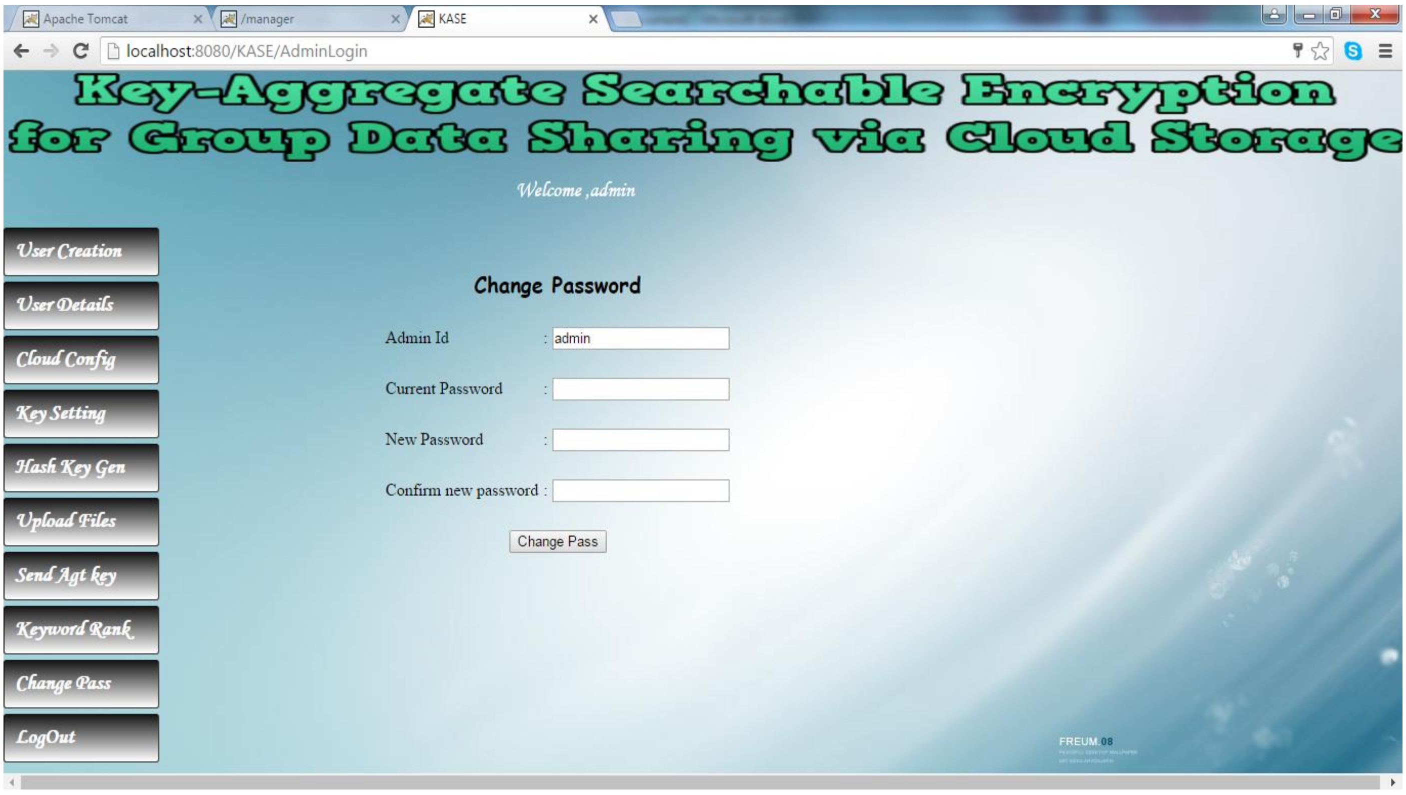




KeyWord Ranking Details

Key No	File Name	Keyword	Weightage	Keyword Rank
1	New Text nnn.txt	sharu	1	9.09
2	New Text nnn.txt	chethu	1	9.09
3	New Text nnn.txt	shri	1	9.09
4	New Text nnn.txt	pallu	1	9.09
5	New Text nnn.txt	manu	1	9.09

« Prev | 1 | 2 | 3 | Next »

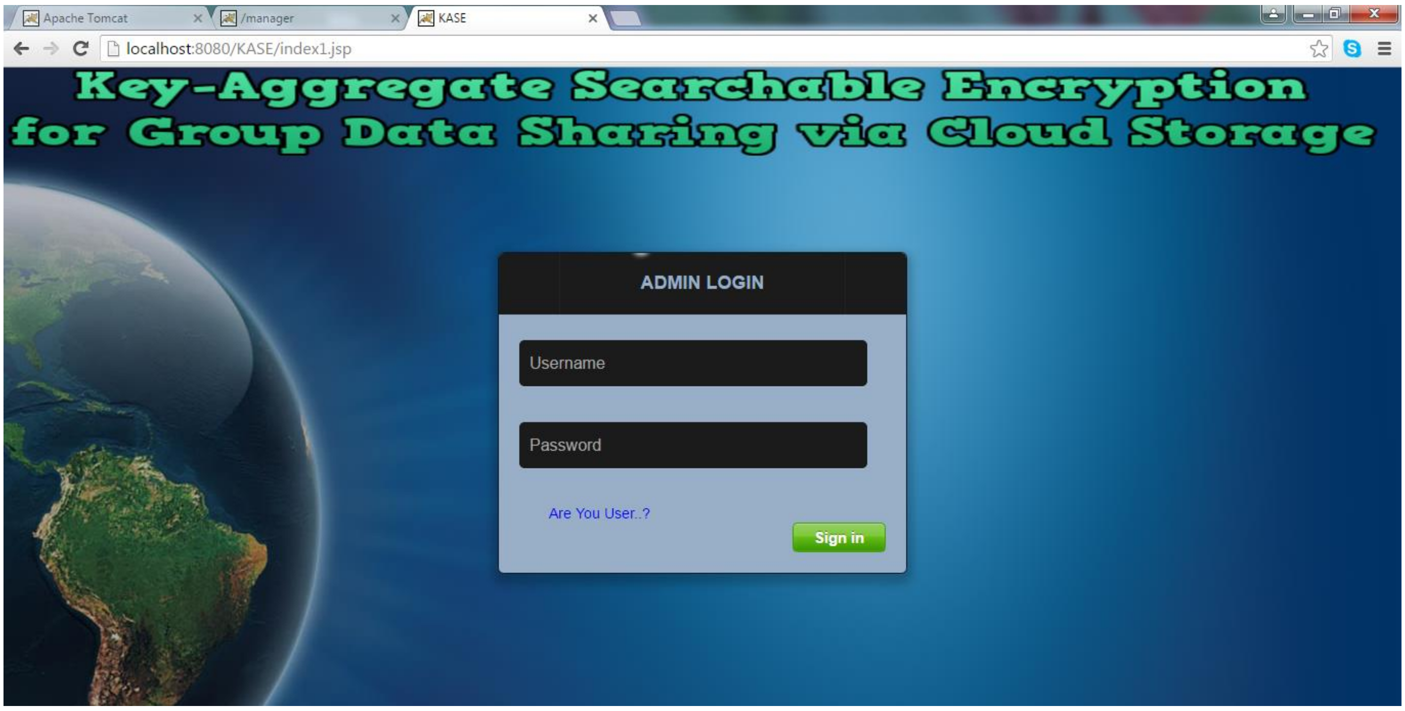


Admin Id : admin

Current Password :

New Password :

Confirm new password :


Apache Tomcat
/manager
KASE
localhost:8080/KASE/index1.jsp
Key-Aggregate Searchable Encryption for Group Data Sharing via Cloud Storage
ADMIN LOGIN
Username
Password
Are You User..?
Sign in