

Crypt Cloud+: Secure and Expressive Data Access Control for Cloud Storage

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering
By

MONISHWAR . S (Reg No. 38290048)

JACOB DANIEL . W (Reg No. 38290027)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING



**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI-600 119, TAMILNADU.
MARCH 2021**



**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC
(Established under section 3 of UGC Act, 1956)
JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI-600 119.**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **MONISHWAR.S (Reg. No. 38290048)** and **JACOB DANIEL.W (Reg. No. 38290027)** who carried out the project entitled “**Crypt Cloud+: Secure and Expressive Data Access Control for Cloud Storage**” under our supervision from November 2020 to March 2021.

Internal Guide
Dr. A.Christy, MCA.,Ph.D.,

Internal Examiner

External Examiner

DECLARATION

I, **MONISHWAR.S (Reg. No. 38290048)** hereby declare that the Project Report entitled “**Crypt Cloud+: Secure and Expressive Data Access Control for Cloud Storage**” done by me under the guidance of **Dr.A.Christy, MCA.,Ph.D.**, is submitted in partial fulfillment of the requirements for the **Award of Bachelor of Science Degree in Computer Science**.

Date:

Place: Chennai

Signature of the candidate

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D.**, Dean, School of Computing and **Dr. S. VIGNESHWARI, M.E., Ph.D.**, and **Dr.L.LAKSHMANAN, M.E., Ph.D.**, Head of the Department, Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.A.Christy, MCA.,Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways For the Completion of the project.

ABSTRACT

Secure cloud storage, which is an emerging cloud service, is designed to protect the confidentiality of outsourced data but also to provide flexible data access for cloud users whose data is out of physical control. Cipher text Policy Attribute-Based Encryption (CP-ABE) is regarded as one of the most promising techniques that may be leveraged to secure the guarantee of the service. However, the use of CP-ABE may yield an inevitable security breach which is known as the misuse of access credential (i.e. decryption rights), due to the intrinsic “all-or-nothing” decryption feature of CP-ABE. In this project, we investigate the two main cases of access credential misuse: one is on the semi-trusted authority side, and the other is on the side of cloud user. To mitigate the misuse, we propose the first accountable authority and revocable CP-ABE based cloud storage system with white-box traceability and auditing, referred to as Crypt Cloud. We also present the security analysis and further demonstrate the utility of our system via experiments.

TABLE OF CONTENTS

CHAPTER No	TITLE	PAGE
1	INTRODUCTION	
1.1	Aim	9
1.2	Synopsis	9
2	SYSTEM ANALYSIS	
2.1	Existing system	10
2.2	Proposed system	10
3	REQUIREMENTS SPECIFICATION	
3.1	Introduction	11
3.2	Hardware and Software specification	12
3.3	Technologies Used	12
3.4	Algorithm	13
4	Objective and Scope	
4.1	Objective	14
4.2	Project Scope	14
4.3	Product Perspective	14

4.4	System Features	15
4.5	Design and Implementation Constraints	16
4.6	Safety Requirement	17
5	SYSTEM DESIGN	
5.1	Architecture Diagram	19
5.2	Sequence Diagram	20
5.3	Usecase Diagram	20
5.4	Activity Diagram	22
5.5	Collaboration Diagram	23
5.6	Data flow Diagram	24
6	SYSTEM DESIGN – DETAILED	
6.1	Modules	26
6.2	Module explanation	27
7	CODING	
7.1	Coding	38
7.2	Coding standards	38
	Future work	41
	Conclusion	41

	Reference	42
	Source code	44

CHAPTER 1

INTRODUCTION

1.1 Aim:

The main aim of this project is to provide integrity of an organization data which is in public cloud.

1.2 Synopsis:

Data owners will store their data in public cloud along with encryption and particular set of attributes to access control on the cloud data. While uploading the data into public cloud they will assign some attribute set to their data. If any authorized cloud user wants to download their data they should enter that particular attribute set to perform further actions on data owner's data. A cloud user wants to register their details under cloud organization to access the data owner's data. Users want to submit their details as attributes along with their designation. Based on the user details Semi-Trusted Authority generates decryption keys to get control on owner's data. An user can perform a lot of operations over the cloud data. If the user wants to read the cloud data he needs to be entering some read related attributes, and if he wants to write the data he needs to be entering write related attributes. For each and every action user in an organization would be verified with their unique attribute set. These attributes would be shared by the admins to the authorized users in cloud organization. These attributes will be stored in the policy files in a cloud. If any user leaks their unique decryption key to any malicious user, data owners trace by sending audit request to auditor and auditor will process the data owners request and concludes that who is the guilty.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In existing system the CP-ABE may help us prevent security breach from outside attackers. But when an insider of the organization is suspected to commit the “crimes” related to the redistribution of decryption rights and the circulation of user information in plain format for illicit financial gains, how could we conclusively determine that the insider is guilty? Is it also possible for us to revoke the compromised access privileges? In addition to the above questions, we have one more which is related to key generation authority. A cloud user’s access credential (i.e., decryption key) is usually issued by a semi-trusted authority based on the attributes the user possesses. How could we guarantee that this particular authority will not (re-)distribute the generated access credentials to others.

2.2 PROPOSED SYSTEM

In this work, we have addressed the challenge of credential leakage in CP-ABE based cloud storage system by designing an accountable authority and revocable Crypt Cloud which supports white-box traceability and auditing (referred to as Crypt Cloud+). This is the first CP-ABE based cloud storage system that simultaneously supports white-box traceability, accountable authority, auditing and effective revocation. Specifically, Crypt Cloud+ allows us to trace and revoke malicious cloud users (leaking credentials). Our approach can be also used in the case where the users’ credentials are redistributed by the semi-trusted authority.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 INTRODUCTION

The prevalence of cloud computing may indirectly incur vulnerability to the confidentiality of outsourced data and the privacy of cloud users. A particular challenge here is on how to guarantee that only authorized users can gain access to the data, which has been outsourced to cloud, at anywhere and anytime. One naive solution is to employ encryption technique on the data prior to uploading to cloud. However, the solution limits further data sharing and processing. This is so because a data owner needs to download the encrypted data from cloud and further re-encrypt them for sharing (suppose the data owner has no local copies of the data). A fine-grained access control over encrypted data is desirable in the context of cloud computing.

Cipher text Policy Attribute-Based Encryption (CPABE) may be an effective solution to guarantee the confidentiality of data and provide fine-grained access control here. In a CP-ABE based cloud storage system, for example, organizations (e.g., a university such as the University of Texas at San Antonio) and individuals (e.g., students, faculty members and visiting scholars of the university) can first specify access policy over attributes of a potential cloud user. Authorized cloud users then are granted access credentials (i.e., decryption keys) corresponding to their attribute sets (e.g., student role, faculty member role, or visitor role), which can be used to obtain access to the outsourced data. As a robust one-to-many encryption mechanism, CP-ABE offers a reliable method to protect data stored in cloud, but also enables fine-grained access control over the data.

3.2 HARDWARE AND SOFTWARE SPECIFICATION

3.2.1 HARDWARE REQUIREMENTS

- Hard Disk : 80GB and Above
- RAM : 4GB and Above
- Processor : P IV and Above

3.2.2 SOFTWARE REQUIREMENTS

- Windows 7 and above
- JDK 1.7
- J2EE
- Tomcat 7.0
- MySQL

3.3 TECHNOLOGIES USED

- J2EE (JSP, Servlets), JavaScript, HTML, CSS, AJAX.

3.3.1 JAVA

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

3.3.2 APACHE TOMCAT SERVER

Apache Tomcat (formerly under the Apache Jakarta Project; Tomcat is now a top level project) is a web container developed at the Apache Software

Foundation. Tomcat implements the servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, providing an environment for Java code to run in cooperation with a web server. It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Because Tomcat includes its own HTTP server internally, it is also considered a standalone web server.

3.4 Algorithm

RSA (Rivest–Shamir–Adleman) is an **algorithm** used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic **algorithm**. Asymmetric means that there are two different keys. This is also called public key cryptography, because one of the keys can be given to anyone.

Binary search

Binary search is a fast **search algorithm** with run-time complexity of $O(\log n)$ For this **algorithm** to work properly, the data collection should be in the sorted form. **Binary search** looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned.

CHAPTER 4

Objective and Scope

4.1 Objective

The main Objective of this project is to provide integrity of an organization data which is in public cloud.

4.2 Project Scope

In this work, we have addressed the challenge of credential leakage in CP-ABE based cloud storage system by designing an accountable authority and revocable Crypt Cloud which supports white-box traceability and auditing (referred to as Crypt Cloud+). This is the first CP-ABE based cloud storage system that simultaneously supports white-box traceability, accountable authority, auditing and effective revocation. Specifically, Crypt Cloud+ allows us to trace and revoke malicious cloud users (leaking credentials). Our approach can be also used in the case where the users' credentials are redistributed by the semi-trusted authority.

4.3 Product Perspective

Data owners will store their data in public cloud along with encryption and particular set of attributes to access control on the cloud data. Cloud owners have all rights to download and delete their data whenever they want. While uploading the data into public cloud they will assign some attribute set to their data. If any authorized cloud user wants to download their data they should enter that particular attribute set to perform further actions on data owner's data. A cloud user wants to register their details under cloud organization to access the data owner's data. Users want to submit their details as attributes along with their designation.

Based on the user details Semi-Trusted Authority generates decryption keys to get control on owner's data. A user can perform a lot of operations over the cloud data. If the user wants to read the cloud data he needs to be entering some read related attributes, and if he wants to write the data he needs to be entering write related attributes. For each and every action user in an organization would be verified with their unique attribute set. These attributes would be shared by the admins to the authorized users in cloud organization. These attributes will be stored in the policy files in a cloud. If any user leaks their unique decryption key to the any malicious user data owners wants to trace by sending audit request to auditor and auditor will process the data owners request and concludes that who is the guilty.

4.4 System Features

- 1) Traceability of malicious cloud users. Users who leak their access credentials can be traced and identified.
- 2) Accountable authority. A semi-trusted authority, who (without proper authorization) generates and further distributes access credentials to unauthorized user(s), can be identified. This allows further actions to be undertaken (e.g. criminal investigation or civil litigation for damages and breach of contract).
- 3) Auditing. An auditor can determine if a (suspected) cloud user is guilty in leaking his/her access credential.

4) “Almost” zero storage requirement for tracing. We use a Paillier-like encryption as an extractable commitment in tracing malicious cloud users and more practically, we do not need to maintain an identity table of users for tracing (unlike the approach used in [27]).

5) Malicious cloud users’ revocation. Access credentials for individual traced and further determined to be “compromised” can be revoked. We design two mechanisms to revoke the “traitor(s)” effectively. The ATER-CP-ABE provides an explicitly revocation mechanism where a revocation list is specified explicitly into the algorithm **Encrypt**, while the ATIRCP- ABE offers an implicitly revocation where the encryption

does not need to know the revocation list but a key update operation is required periodically

4.5 Design and Implementation Constraints

4.5.1 Constraints in Analysis

- ◆ Constraints as Informal Text
- ◆ Constraints as Operational Restrictions
- ◆ Constraints Integrated in Existing Model Concepts
- ◆ Constraints as a Separate Concept
- ◆ Constraints Implied by the Model Structure

4.5.2 Constraints in Design

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

4.5.3 Constraints in Implementation

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

4.6 Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.
3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.
4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.

5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.
6. Systems with different requirements for safety levels must be separated.
7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 5

5.1 Architecture Diagram:

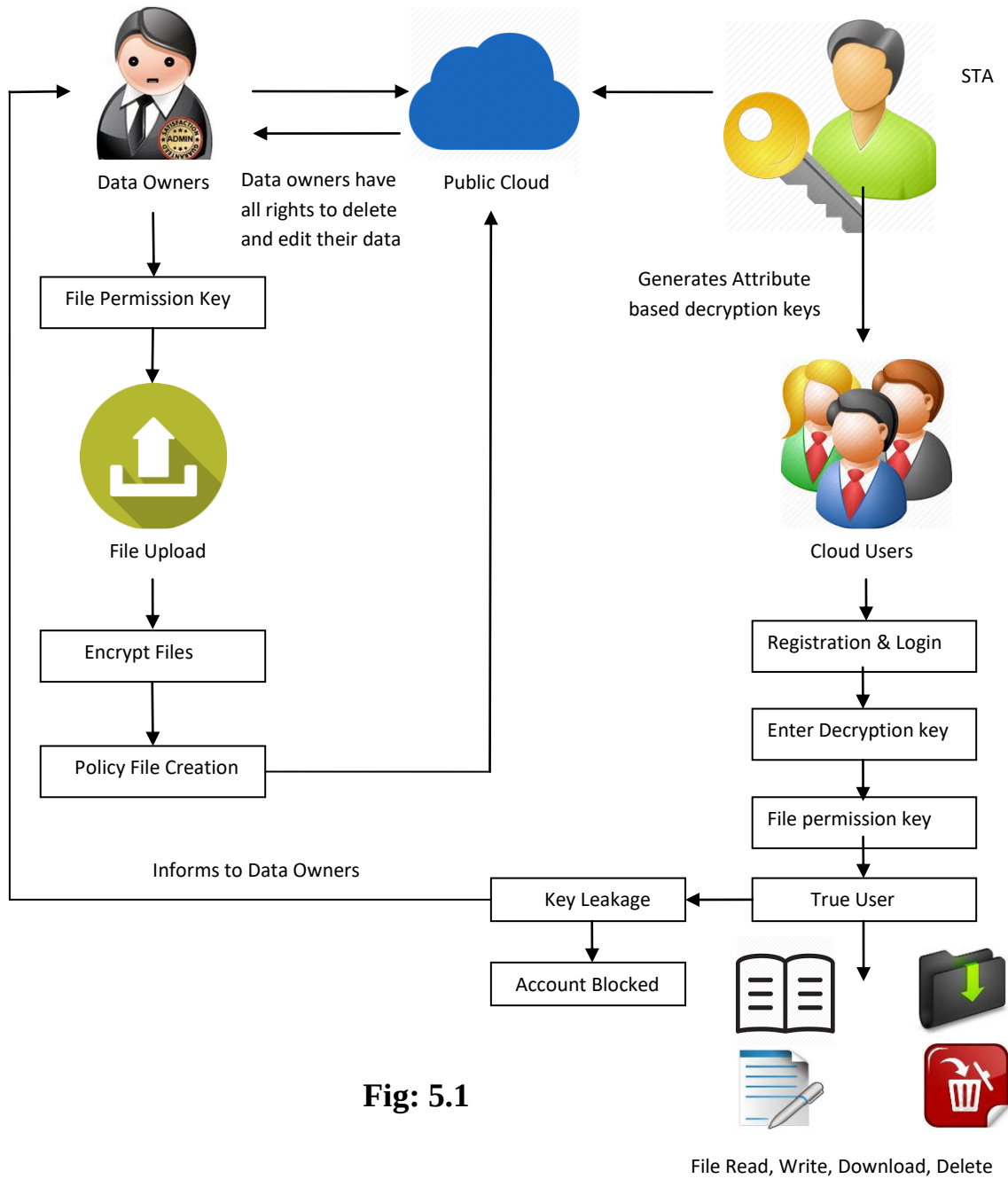


Fig: 5.1

5.2 Sequence Diagram:

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

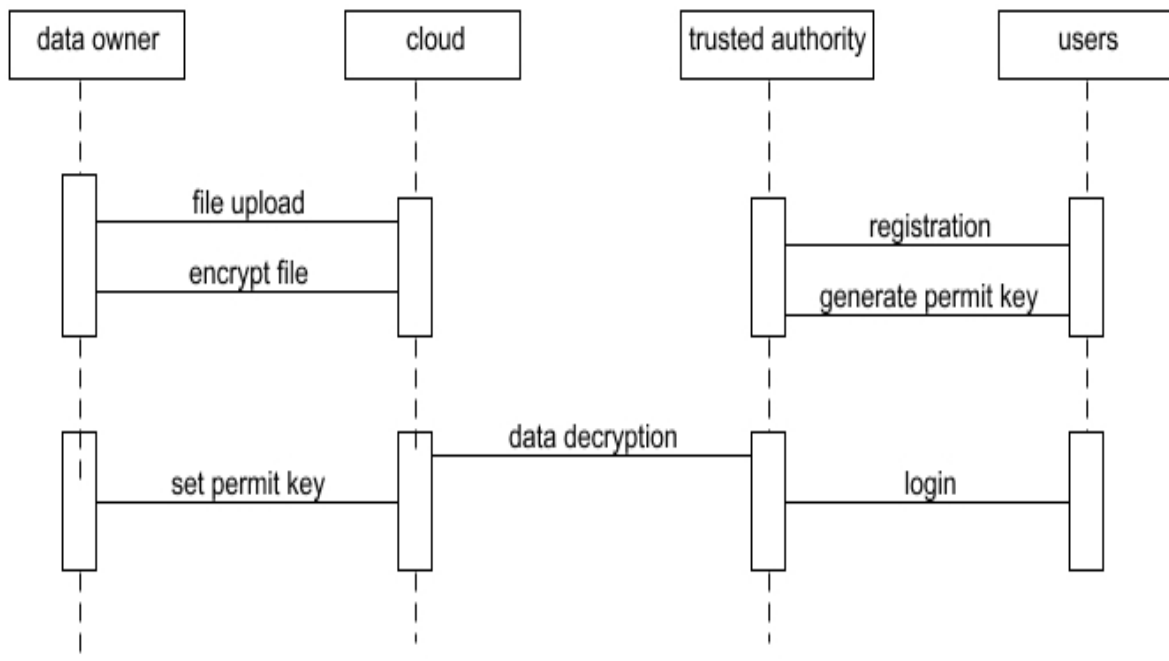


Fig:5.2

5.3 Use Case Diagram:

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This

language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

5.3.1. USECASE DIAGRAM

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

Use case diagram consists of two parts:

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

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

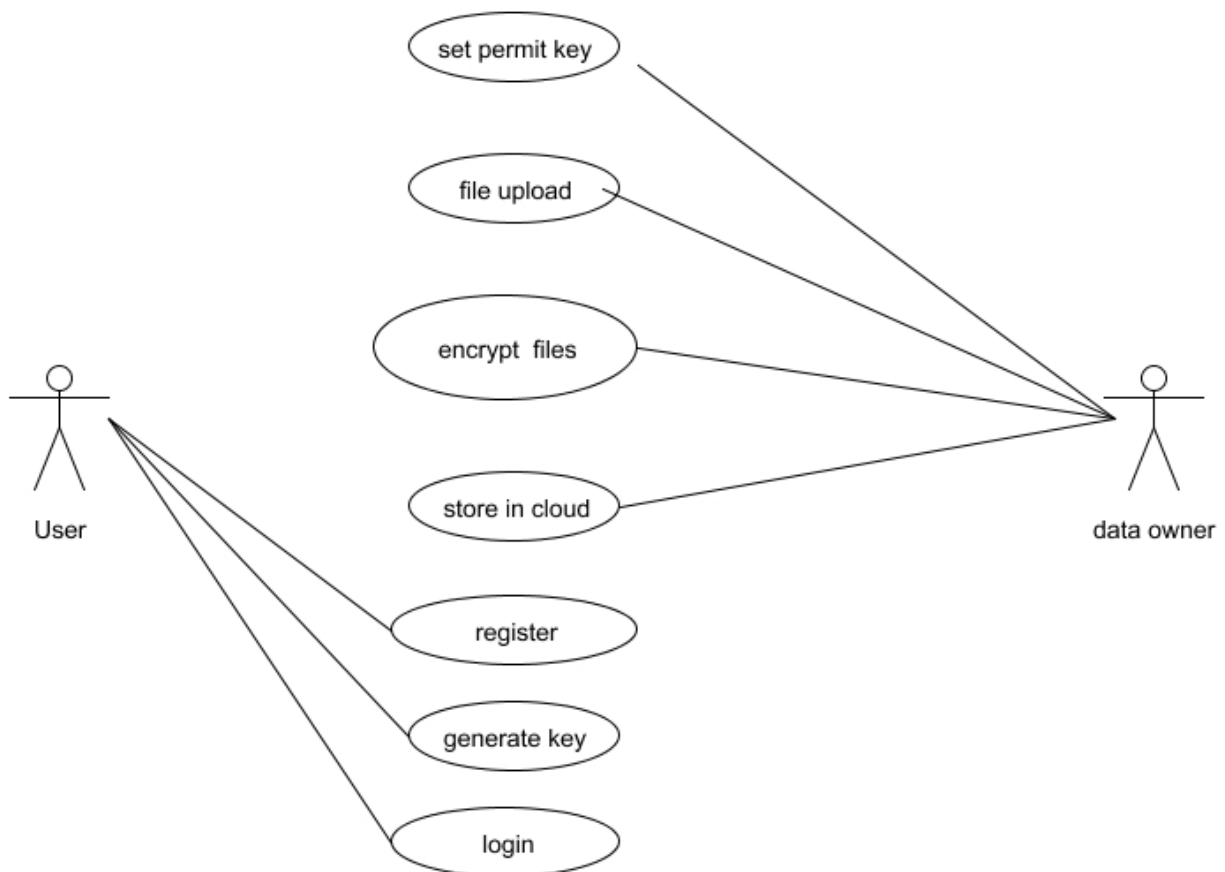


Fig:5.3

5.4 Activity Diagram:

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

The most important shape types:

- Rounded rectangles represent activities.
- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.

- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

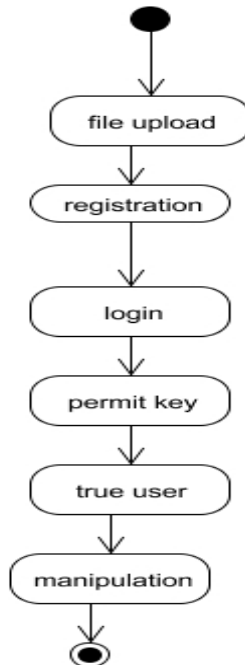


Fig:5.4

5.5 Collaboration Diagram:

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

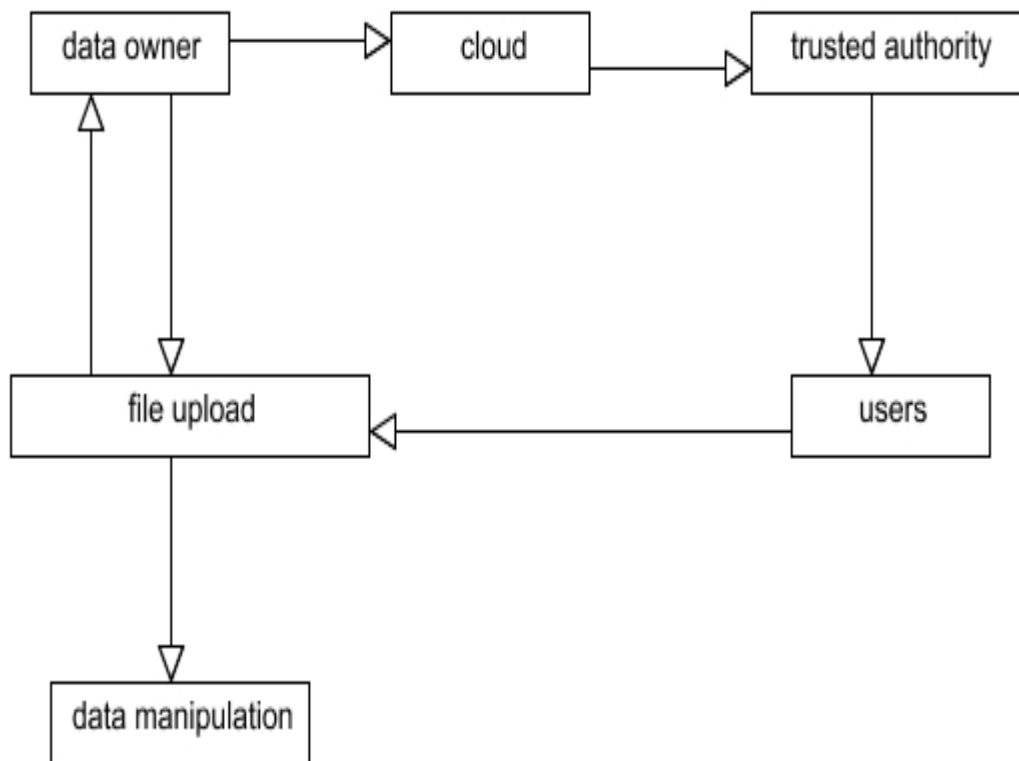


Fig:5.5

5.6 DATA FLOW DIAGRAM:

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

Level 0:

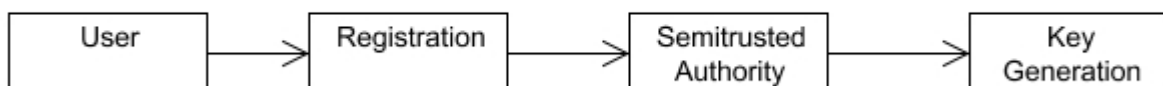


Fig:5.6

Level 1:

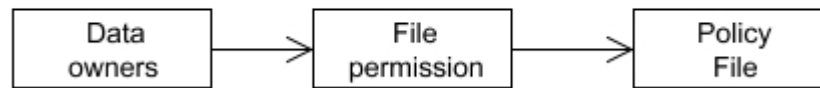


Fig:5.7

Level 2:



Fig:5.8

Level 3

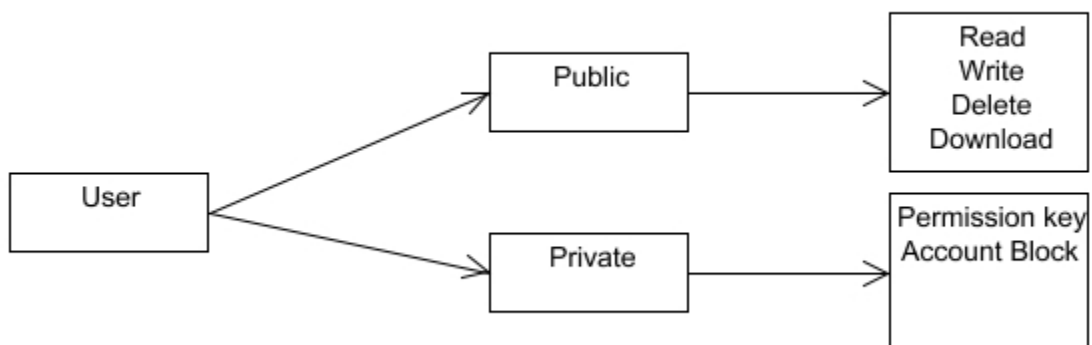


Fig:5.9

CHAPTER 6

SYSTEM DESIGN

Data owners (DOs) encrypt their data under the relevant access policies prior to outsourcing the (encrypted) data to a public cloud (PC). PC stores the outsourced (encrypted) data from Dos and handles data access requests from data users (DUs) Authorized DUs are able to access (e.g. download and decrypt) the outsourced data. Semi-trusted authority (AT) generates system parameters and issues access credentials (i.e., decryption keys) to DUs. Auditor (AU) is trusted by other entities; takes charge of audit and revoke procedures, and returns the trace and audit results to DOs and DUs.

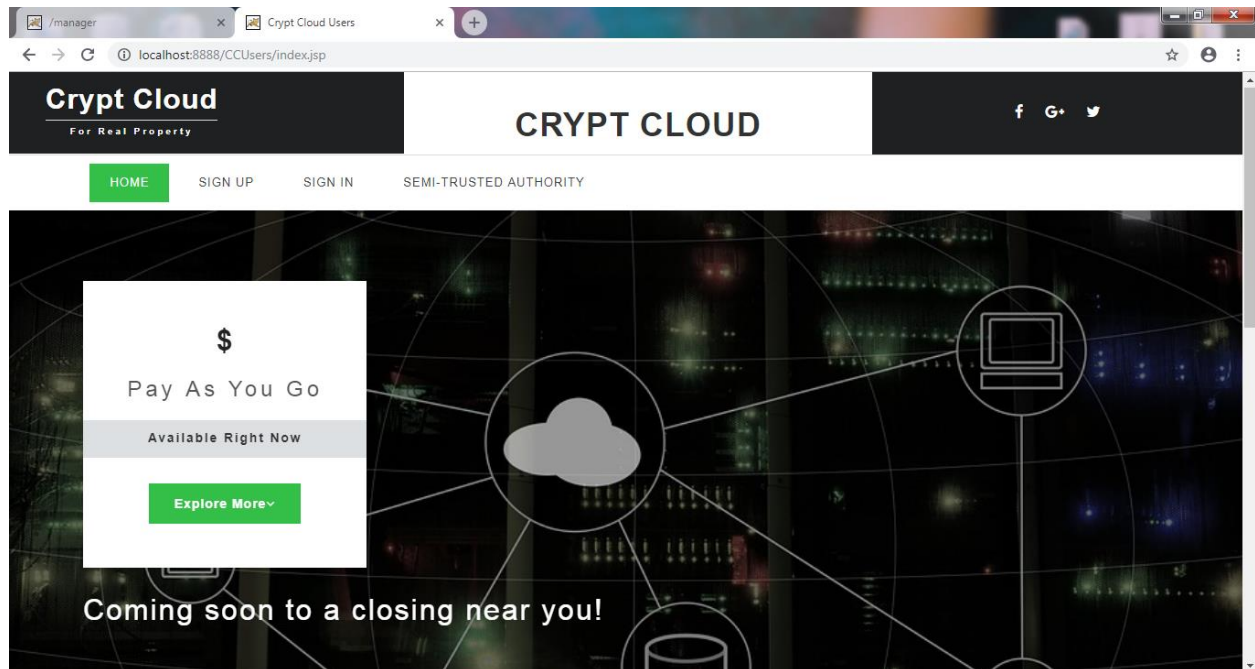


Fig:6.1

6.1 MODULES

- Organization profile creation & Key Generation
- Data Owners File Upload

- **File Permission & Policy File Creation**
- **Tracing who is guilty**

6.2 MODULE EXPLANATION:

6.2.1 Organization profile creation & Key Generation

User has an initial level Registration Process at the web end. The users provide their own personal information for this process.

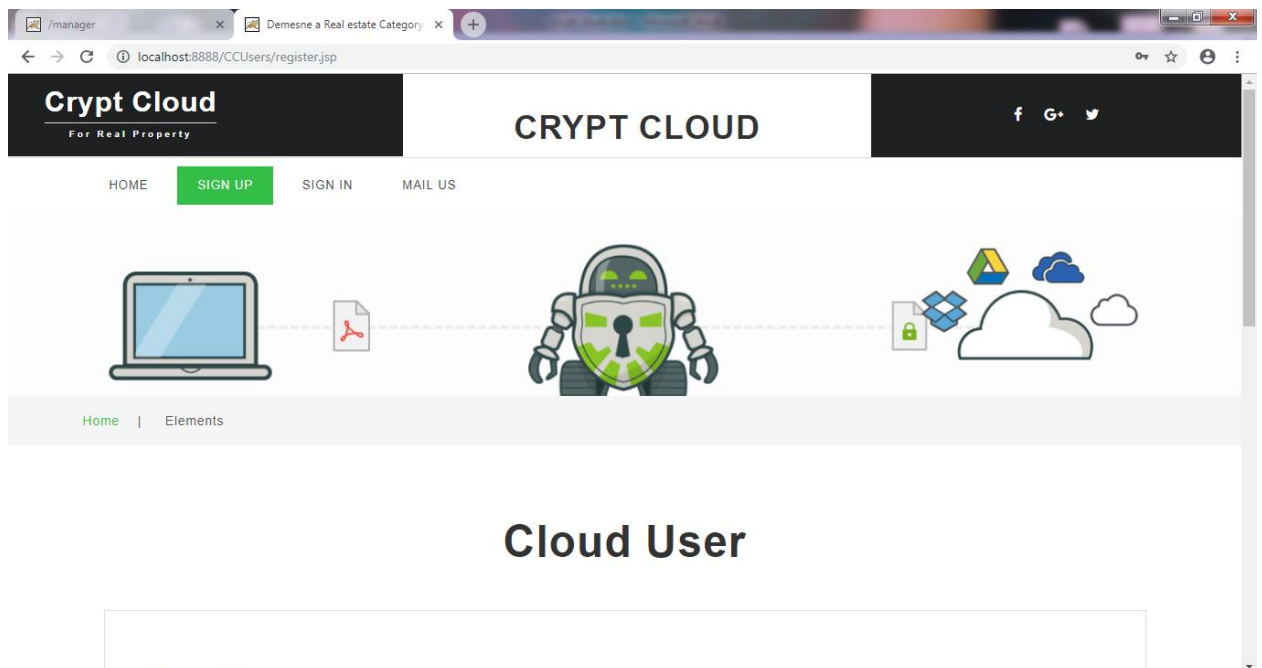


Fig:6.2

User Sign up

Project Manager

Nitishkumar Reddy

nitishkumarreddy.b@gmail.com

9133091999

Chennai

Tamil Nadu

India

Sign Up

Fig:6.3

The server in turn stores the information in its database. Now the Accountable STA (semi-trusted Authority) generates decryption keys to the users based on their Attributes Set (e.g. name, mail-id, contact number etc.,). User gets the provenance to access the Organization data after getting decryption keys from Accountable STA.

Semi-Trusted Authority

STA Login

Username

Password

Signin

Fig:6.4

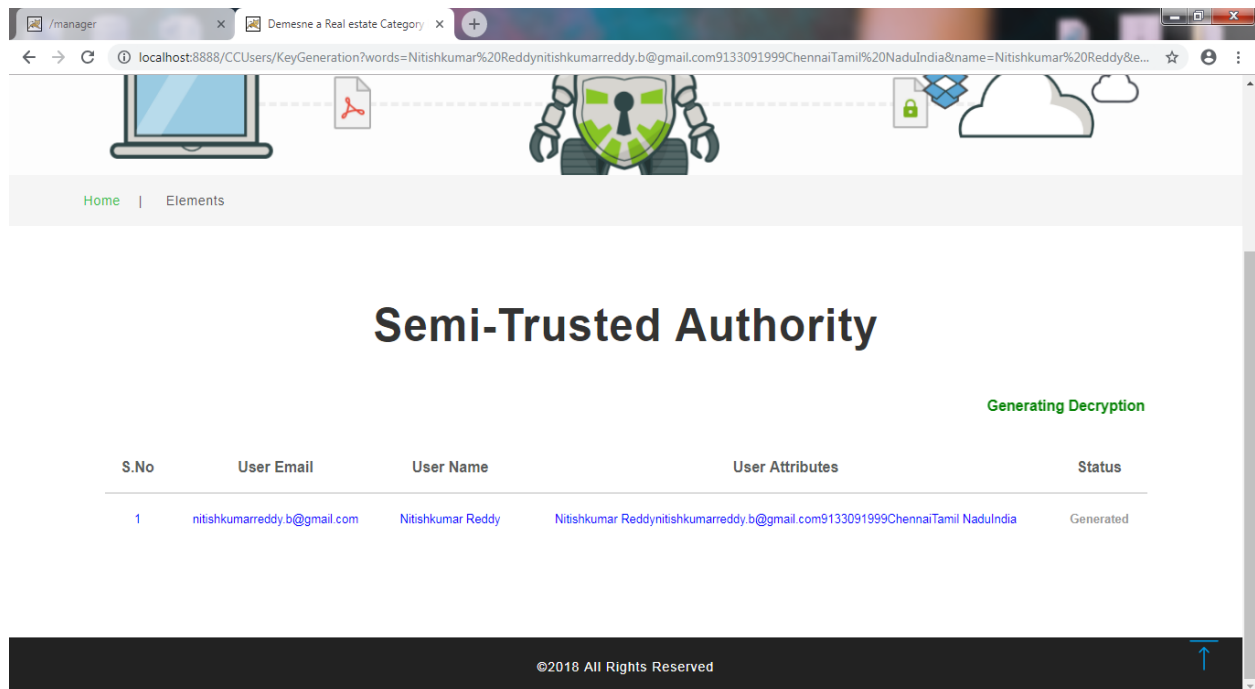


Fig:6.5

6.2.2 Data Owners File Upload

In this module data owners create their accounts under the public cloud and upload their data into public cloud. While uploading the files into public cloud data owners will encrypt their data using RSA Encryption algorithm and generates public key and secret key. And also generates one unique file access permission key for the users under the organization to access their data.

The screenshot shows a web browser with three tabs: /manager, Crypt Cloud Users, and Data Owners. The address bar shows the URL localhost:8888/CCDataOwners/register.jsp. The page has a header with the 'Data Owners' logo and a 'Sign In / Register' button. The main content area is titled 'Register Here' with a subtitle 'Having hands on experience in creating innovative designs, I do offer design solutions which harness'. Below this is a 'Personal Information' section with three input fields: 'Data Owner Name*' containing 'a', 'Email Address*' containing 'a@gmail.com', and 'Password*' which is empty. A red 'Get It' button is at the bottom of the form.

Data Owners

Sign In / Register

Register Here

Having hands on experience in creating innovative designs, I do offer design solutions which harness

Personal Information

Data Owner Name* : a

Email Address* : a@gmail.com

Password* :

Get It

Fig:6.6

The screenshot shows the same web browser with the address bar displaying localhost:8888/CCDataOwners/fileupload.jsp. The page is titled 'File Upload' and includes a welcome message 'Welcome a...!!'. On the left is an image of two people working at a computer. On the right, there is a section titled 'Please upload your file here' with a 'Choose File' button, a text input field containing 'demo.txt', and an 'Upload' button. The footer contains the text '© 2018 All rights reserved'.

File Upload

Welcome a...!!

Please upload your file here

Choose File demo.txt

Upload

© 2018 All rights reserved

Fig:6.7

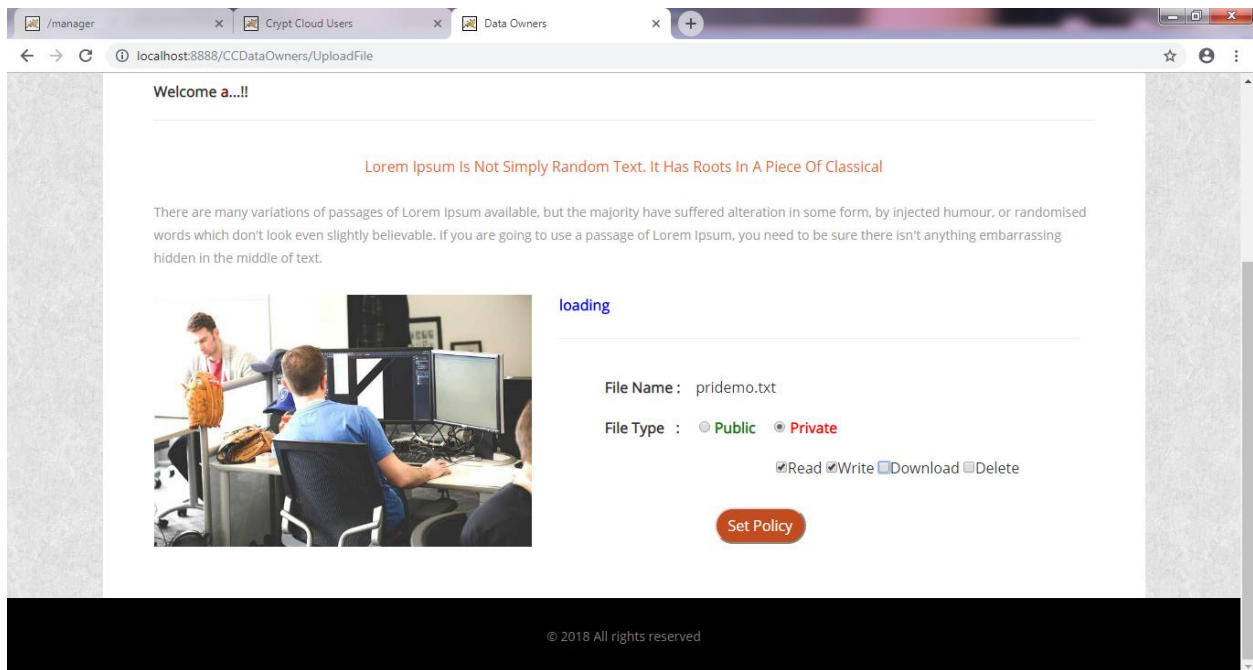


Fig 6.8

6.2.3 File Permission & Policy File Creation

Different data owners will generate different file permission keys to their files and issues those keys to users under the organization to access their files. And also generates policy files to their data that who can access their data. Policy File will split the key for read the file, write the file, download the file and delete the file.

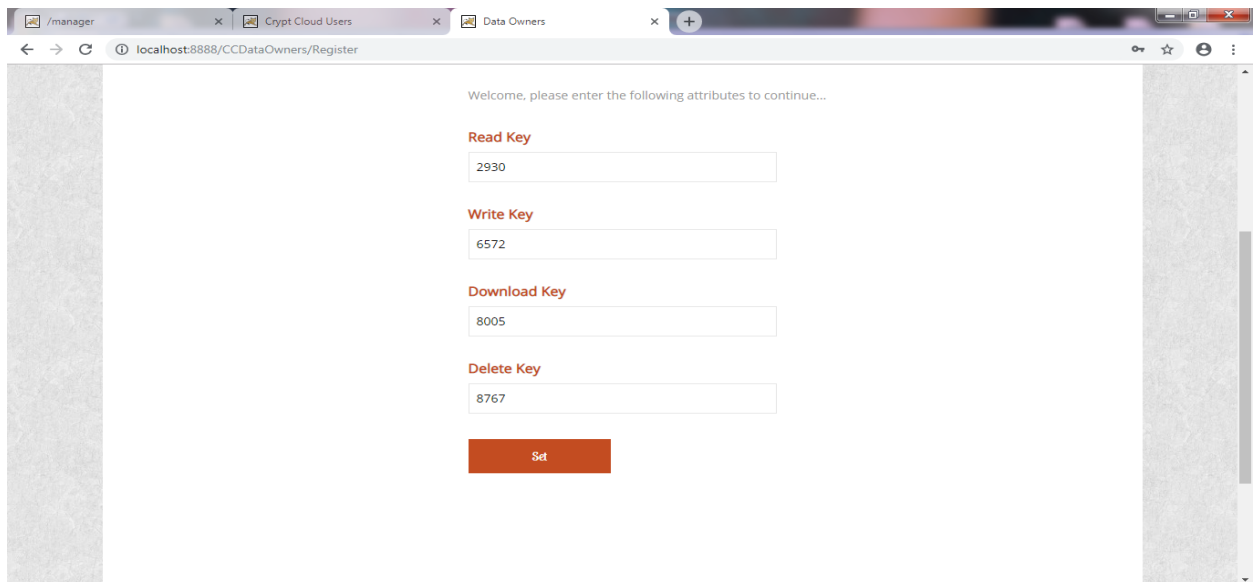


Fig:6.9

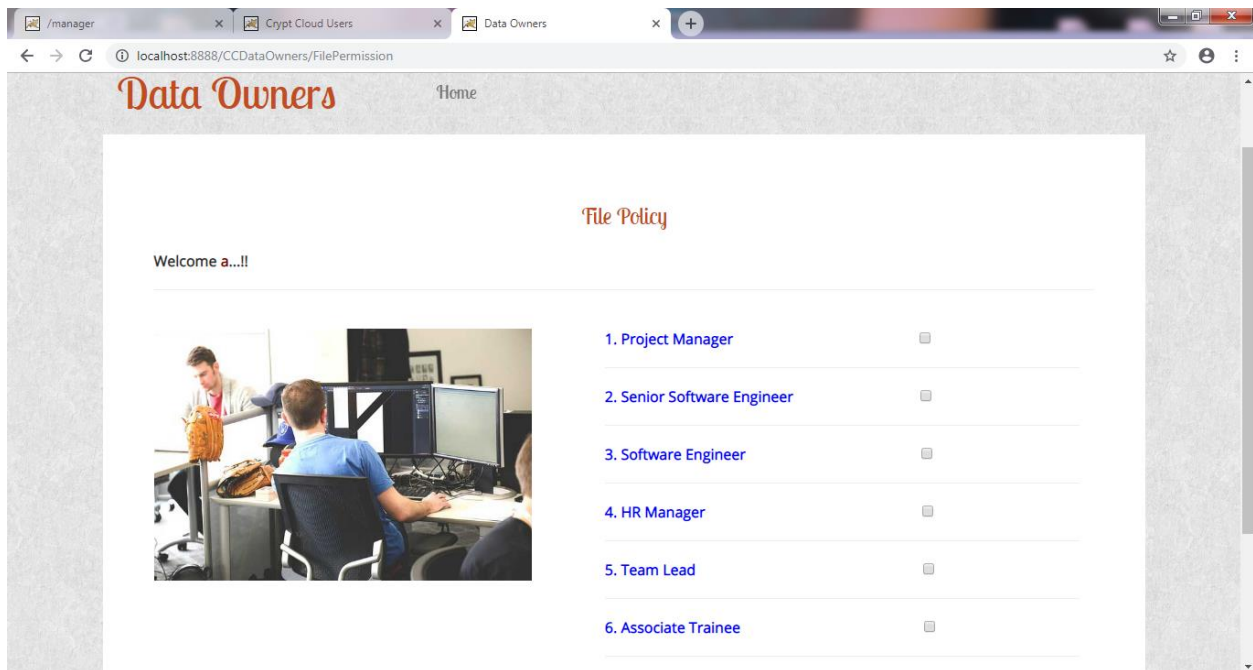


Fig:6.10

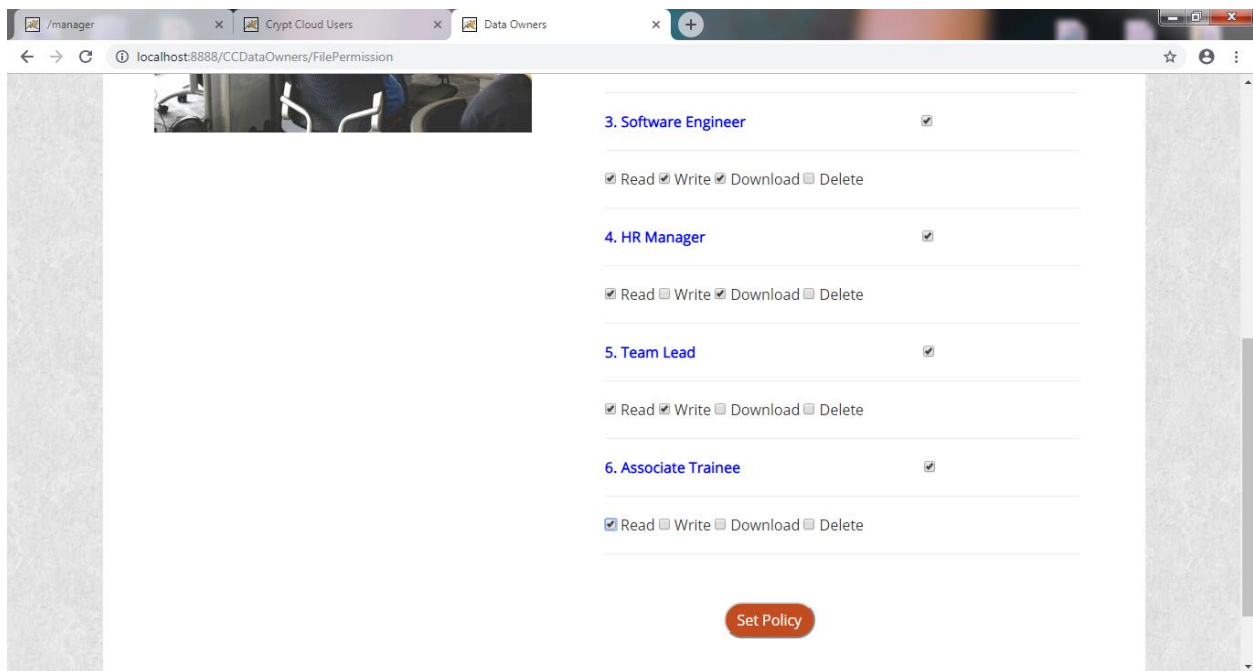


Fig:6.11

6.2.4 Tracing who is guilty

Authorized DUs are able to access (e.g. read, write, download, delete and decrypt) the outsourced data. Here file permission keys are issued to the employees

in the organization based on their experience and position. Senior Employees have all the permission to access the files (read, write, delete, & download). Fresher's only having the permission to read the files. Some Employees have the permission to read and write. And some employees have all the permissions except delete the data.

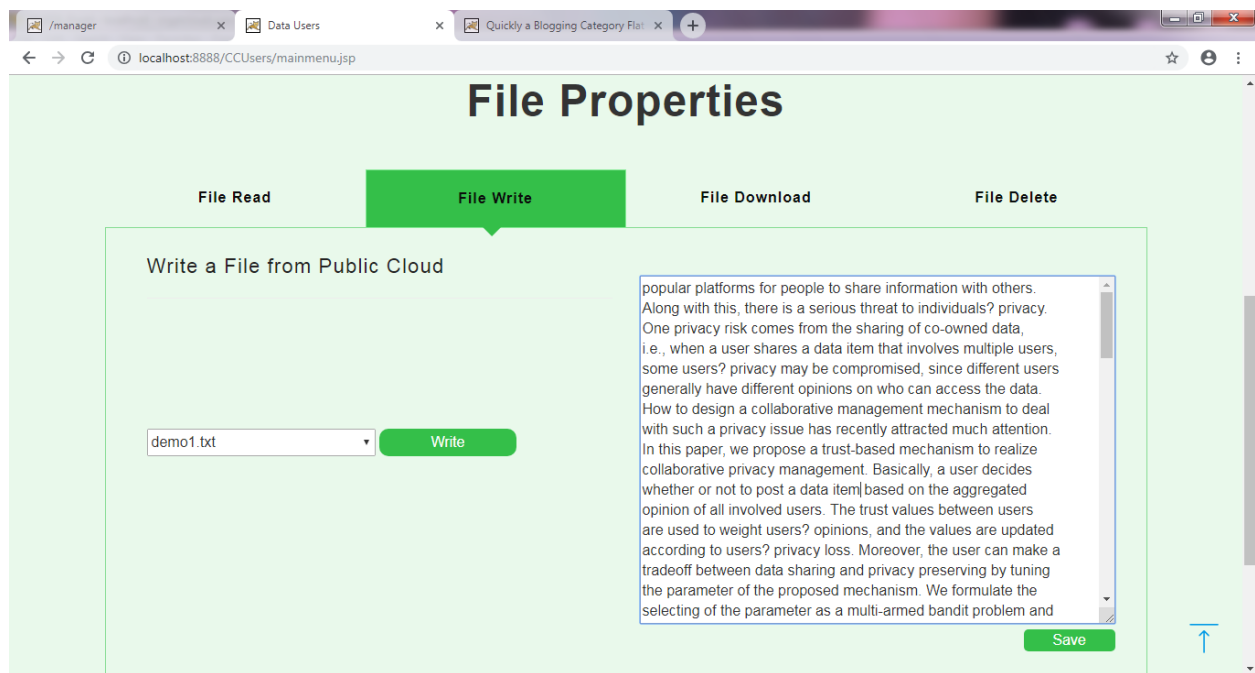


Fig:6.12

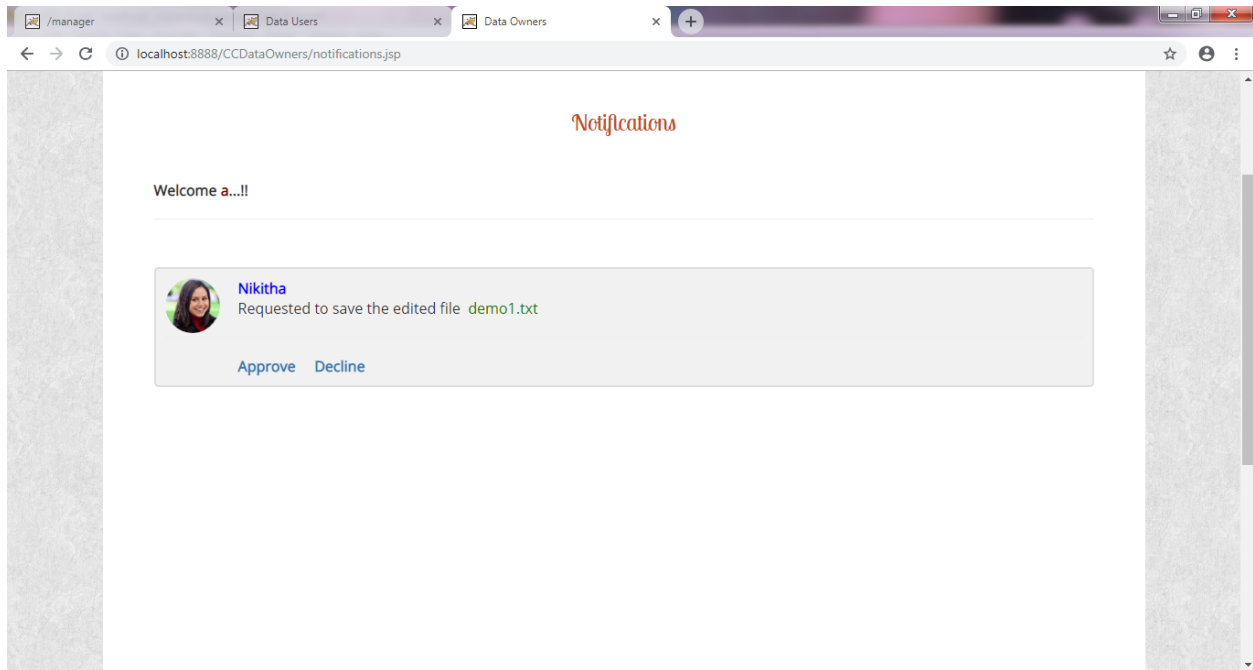


Fig:6.13

If any Senior Employee leaks or shares their secret permission keys to their junior employees they will request to download or delete the Data Owners Data. While entering the key system will generate attribute set for their role in background validate that the user has all rights to access the data. If the attributes set is not matched to the Data Owners policy files they will be claimed as guilty. If we ask them we will find who leaked the key to the junior employees.

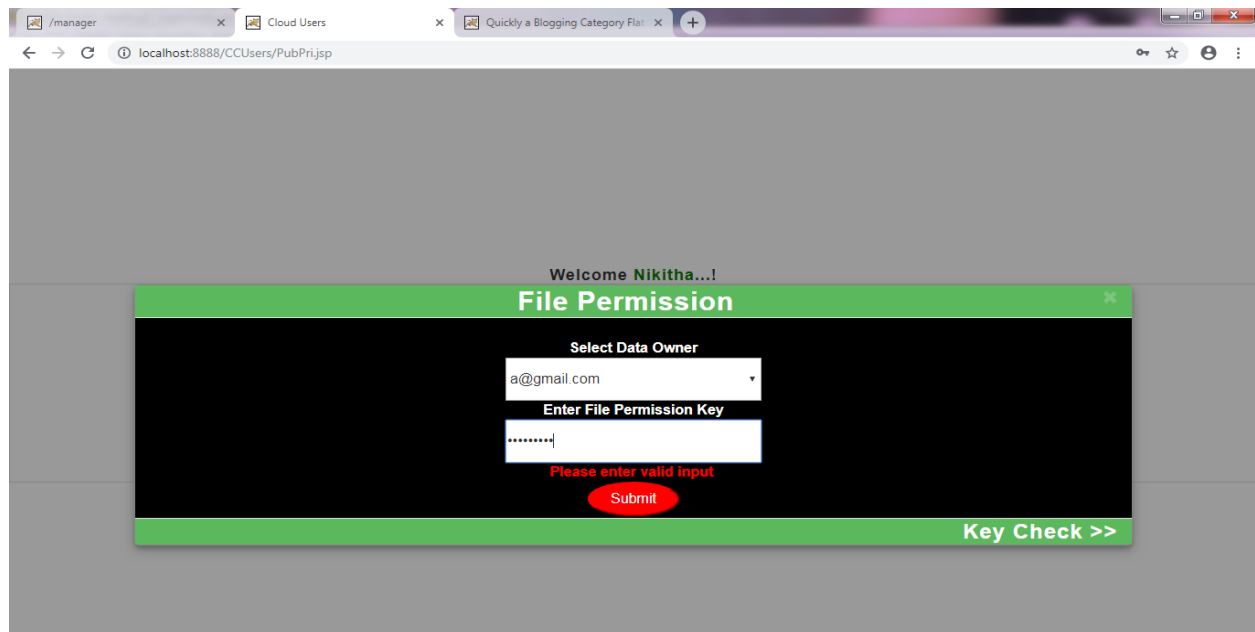


Fig:6.14

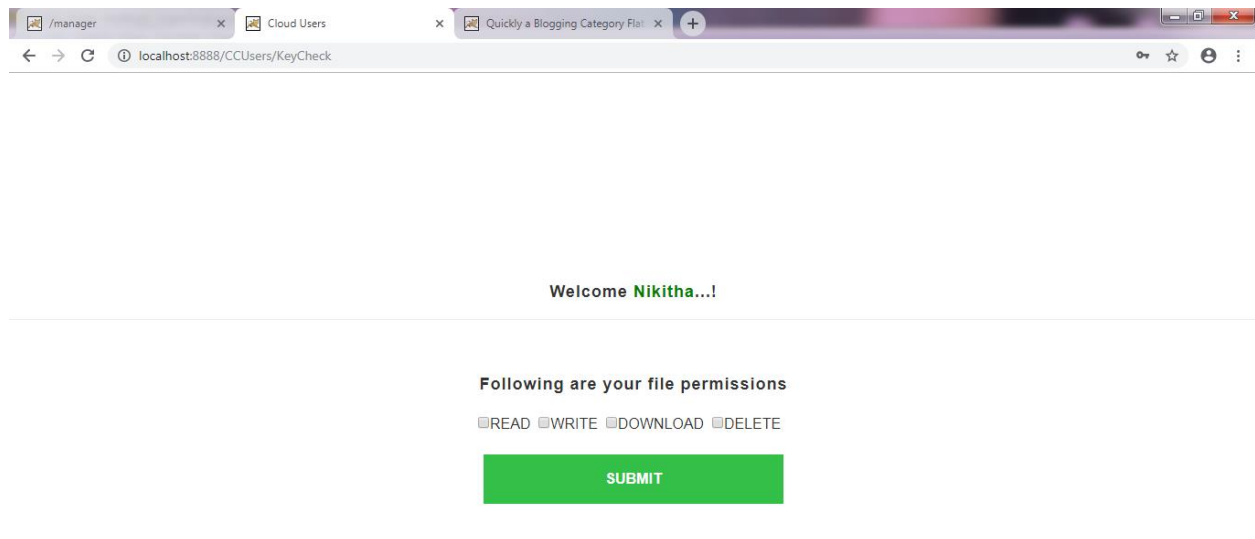


Fig:6.15

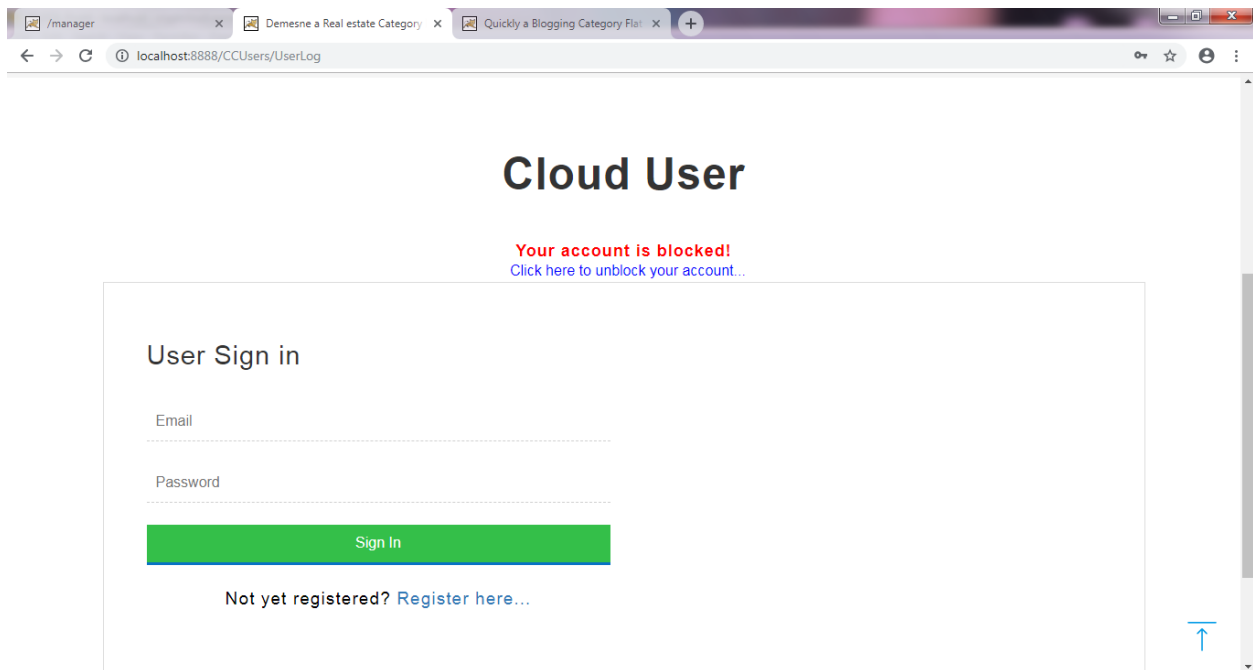


Fig:6.16

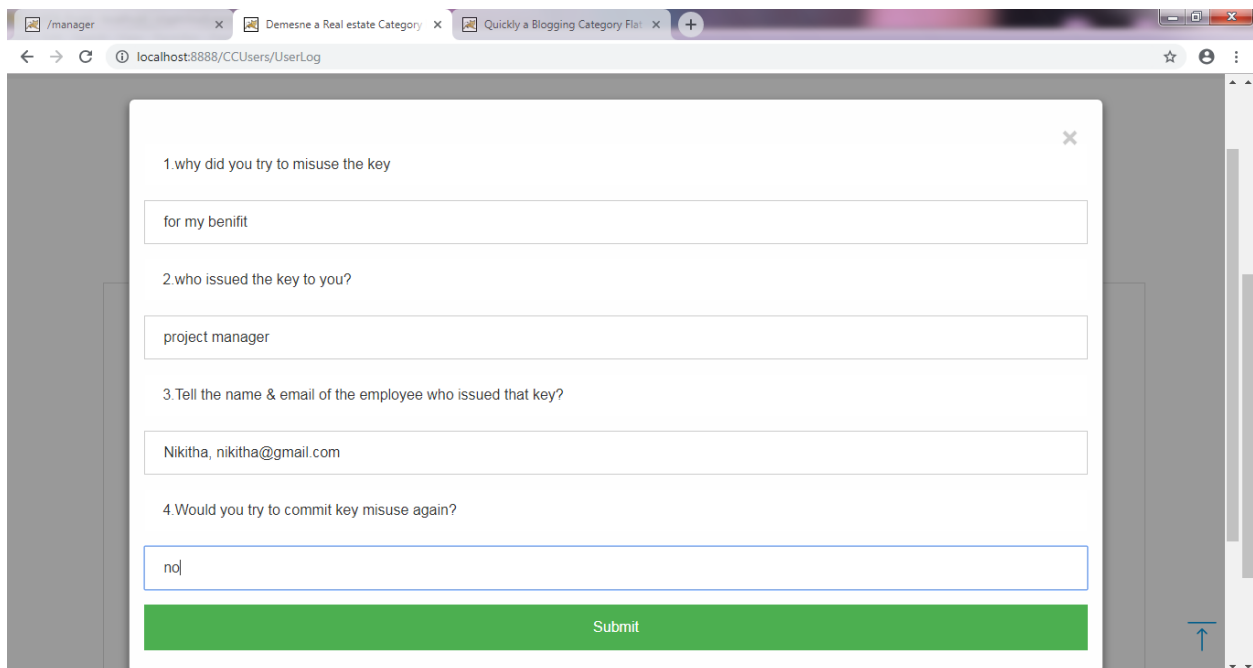


Fig:6.17

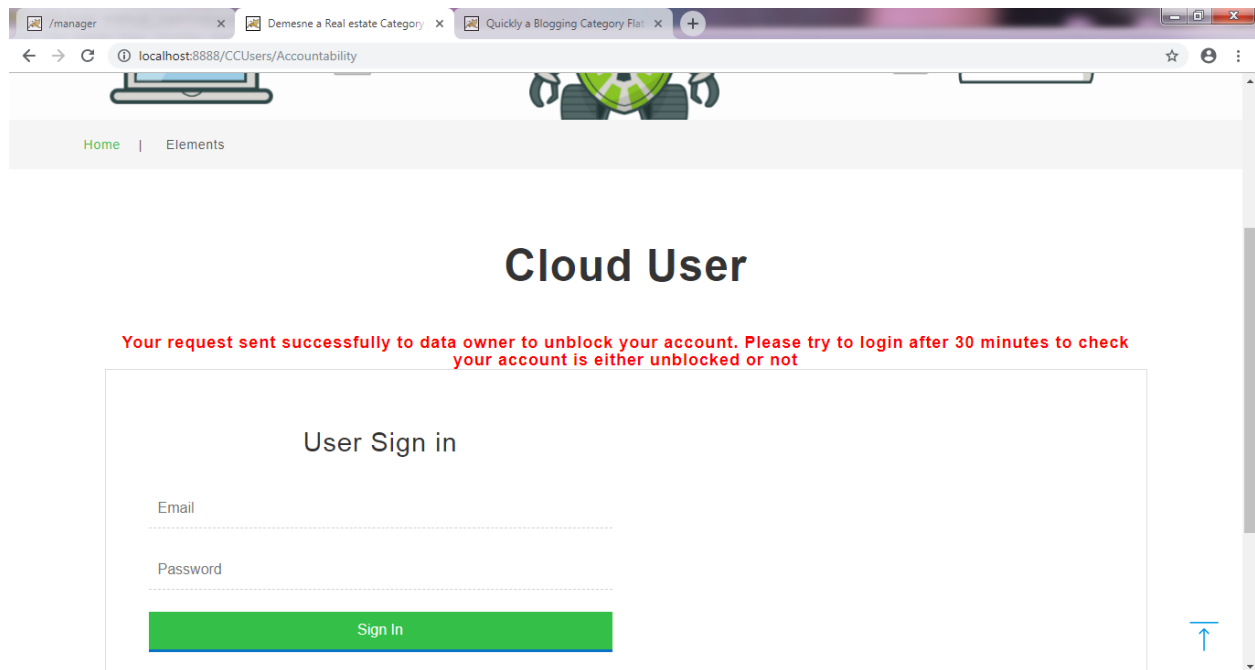


Fig:6.18

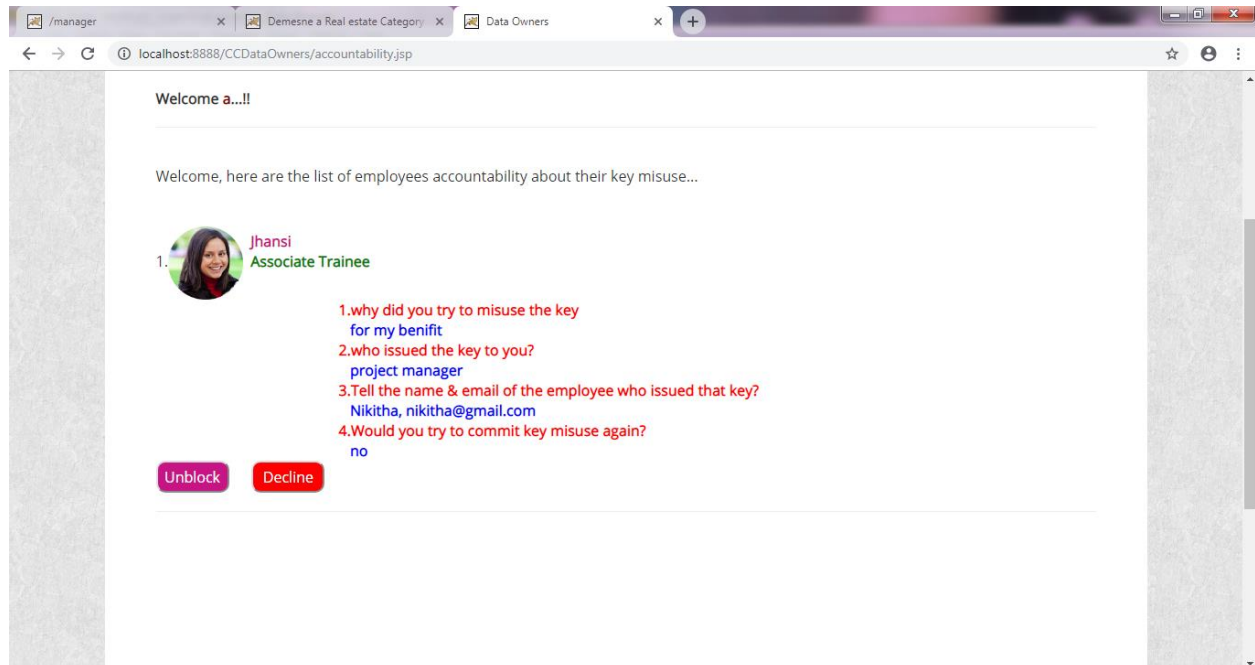


Fig:6.19

CHAPTER 7

CODING

7.1 CODING

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

7.2 CODING STANDARDS

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

Program should be simple, clear and easy to understand.

Naming conventions

Value conventions

Script and comment procedure

Message box format

Exception and error handling

7.2.1 NAMING CONVENTIONS

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

Class names

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

Member Function and Data Member name

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

7.2.2 VALUE CONVENTIONS

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

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

7.2.3 SCRIPT WRITING AND COMMENTING STANDARD

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

7.2.4 MESSAGE BOX FORMAT

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

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

Future work

Our future work will include extending CryptCloud+ to provide “partial” and fully public traceability without compromising on performance.

Conclusion

In this work, we have addressed the challenge of credential leakage in CP-ABE based cloud storage system by designing an accountable authority and revocable CryptCloud which supports white-box traceability and auditing (referred to as CryptCloud+). This is the first CP-ABE based cloud storage system that simultaneously supports white-box traceability, accountable authority, auditing and effective revocation. Specifically, CryptCloud+ allows us to trace and revoke malicious cloud users (leaking credentials). Our approach can be also used in the case where the users’ credentials are redistributed by the semi-trusted authority. We note that we may need black-box traceability, which is a stronger notion (compared to white-box traceability), in CryptCloud. One of our future works is to consider the black-box traceability and auditing.

REFERENCES

1. Mazhar Ali, Revathi Dhamotharan, Eraj Khan, Samee U. Khan, Athanasios V. Vasilakos, Keqin Li, and Albert Y. Zomaya. Sedasc: Secure data sharing in clouds. *IEEE Systems Journal*, 11(2):395–404, 2017.
2. Mazhar Ali, Samee U. Khan, and Athanasios V. Vasilakos. Security in cloud computing: Opportunities and challenges. *Inf. Sci.*, 305:357–383, 2015.
3. Michael Armbrust, Armando Fox, R ean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.
4. Nuttapong Attrapadung and Hideki Imai. Attribute-based encryption supporting direct/indirect revocation modes. In *Cryptography and Coding*, pages 278–300. Springer, 2009.
5. Amos Beimel. Secure schemes for secret sharing and key distribution. PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
6. Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In *Advances in Cryptology-CRYPTO’92*, pages 390–420. Springer, 1993.

7. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In EUROCRYPT - 2004, pages 56–73, 2004.
8. Hongming Cai, Boyi Xu, Lihong Jiang, and Athanasios V. Vasilakos. Iot-based big data storage systems in cloud computing: Perspectives and challenges. IEEE Internet of Things Journal, 4(1):75–87, 2017.
9. Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Advances in Cryptology - EUROCRYPT 2015, pages 595–624, 2015.
10. Angelo De Caro and Vincenzo Iovino. jpbcc: Java pairing based cryptography. In ISCC 2011, pages 850–855. IEEE, 2011.
11. Hua Deng, Qianhong Wu, Bo Qin, Jian Mao, Xiao Liu, Lei Zhang, and Wenchang Shi. Who is touching my cloud. In Computer Security-ESORICS 2014, pages 362–379. Springer, 2014.

Source Code

AccountabilityPojo.java
package logics;

```
public class AccountabilityPojo {  
    private  
    email,question1,answer1,question2,answer2,question3,answer3,question4,answer4,owner;  
    String  
  
    public String getOwner() {  
        return owner;  
    }  
  
    public void setOwner(String owner) {  
        this.owner = owner;  
    }  
  
    public String getEmail() {  
        return email;  
    }  
  
    public void setEmail(String email) {  
        this.email = email;  
    }  
  
    public String getQuestion1() {  
        return question1;  
    }  
  
    public void setQuestion1(String question1) {  
        this.question1 = question1;  
    }  
  
    public String getAnswer1() {  
        return answer1;  
    }  
  
    public void setAnswer1(String answer1) {  
        this.answer1 = answer1;  
    }  
  
    public String getQuestion2() {  
        return question2;  
    }  
  
    public void setQuestion2(String question2) {  
        this.question2 = question2;  
    }  
}
```

```

    }

    public String getAnswer2() {
        return answer2;
    }

    public void setAnswer2(String answer2) {
        this.answer2 = answer2;
    }

    public String getQuestion3() {
        return question3;
    }

    public void setQuestion3(String question3) {
        this.question3 = question3;
    }

    public String getAnswer3() {
        return answer3;
    }

    public void setAnswer3(String answer3) {
        this.answer3 = answer3;
    }

    public String getQuestion4() {
        return question4;
    }

    public void setQuestion4(String question4) {
        this.question4 = question4;
    }

    public String getAnswer4() {
        return answer4;
    }

    public void setAnswer4(String answer4) {
        this.answer4 = answer4;
    }

}

```

```

Decrypt.java
package logics;

```

```

import java.io.IOException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;

import javax.crypto.BadPaddingException;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;

public class Decrypt
{
    private static String passWord1="";
    private static SecretKeyFactory keyFactory;
    private static byte[] passByte;
    Cipher desCipher;
    SecretKey myDesKey ;
    public String decrypt(String cipher,String passWord1) throws
InvalidKeySpecException, NoSuchAlgorithmException, NoSuchPaddingException,
IOException
    {
        String dec="";
        this.passWord1=passWord1;
        try
        {
            manageKeystrengthMethod();
            keyFactory = SecretKeyFactory.getInstance("DES");
            passByte=this.passWord1.getBytes();
            DESKeySpec dspec= new DESKeySpec(passByte);
            SecretKey myDesKey = keyFactory.generateSecret(dspec);
            Cipher desCipher;

            // Create the cipher
            desCipher = Cipher.getInstance("DES/ECB/PKCS5Padding");

            // Initialize the cipher for encryption
            desCipher.init(Cipher.DECRYPT_MODE, myDesKey);

            //sensitive information
            byte[] textEncrypted = cipher.getBytes();

```

```

        // System.out.println("Text [Byte Format] : " + text);
        //System.out.println("Cipher to be decrypted : " + new String(textEncrypted));

        // Decrypt the text
        Base64 bs=new Base64();
        byte[] textDecrypted = desCipher.doFinal(bs.decode(cipher));

        //System.out.println("Text Decrypted : " + new String(textDecrypted));
        dec=new String(textDecrypted);

        }catch(InvalidKeyException e){
            e.printStackTrace();
        }catch(IllegalBlockSizeException e){
            e.printStackTrace();
        }catch(BadPaddingException e){
        }
        return dec;
    }

    private void manageKeystrengthMethod()
    {
        if(passWord1.length()<8)
        {
            int counter=passWord1.length();
            while(counter<8)
            {
                passWord1+='@';
                counter++;
            }
        }
    }
}
EmpChanges.java
package logics;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.http.servlet.HttpsServlet;

public class EmpChanges extends HttpsServlet {

```

```

public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String name=request.getParameter("name");
    String email=request.getParameter("email");
    String hemail=request.getParameter("hemail");
    String desig=request.getParameter("desig");
    String mobile=request.getParameter("mobile");

    int status=FileDao.editEmployee(name, email, desig, mobile, hemail);
    if(status>0)
    {
        System.out.println("Employee details changed in a database");
        request.setAttribute("msg", "Employee details changed in a database");
        RequestDispatcher rd=request.getRequestDispatcher("employees.jsp");
        rd.forward(request, response);
    }
    else
    {
        System.out.println("Employee details not changed in a database");
        request.setAttribute("msg", "Employee details not changed in a database");
        RequestDispatcher rd=request.getRequestDispatcher("employees.jsp");
        rd.forward(request, response);
    }
    out.close();
}
}

```