

CRYPTOGRAPHY ENFORCED DYNAMIC ACCESS CONTROL IN THE CLOUD

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

by

HARINATH PODILA (Reg.no.37110268)

HITHESH JALAPARTHI

(Reg.no.37110275)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the Bonafide work of **HARINATH PODILA (Reg.no.37110268)** and **HITHESHJALAPARTHI (Reg.no.37110275)** who carried out the project entitled "**CRYPTOGRAPHY ENFORCED DYNAMIC ACCESS CONTROL IN THE CLOUD**" under my supervision from **August2020toMarch2021**.

Internal Guide

Mrs. M.D. ANTO PRAVEENA, M.C.A., M.E., (Ph.D.),

Head of the Department

Dr.S.VIGNESHWARI M.E.,Ph.D.,
Dr.L.LAKSHMANAN M.E.,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **HARINATH PODILA, HITHESH JALAPARTHI** hereby declare that the Project Report entitled “**CRYPTOGRAPHY ENFORCED DYNAMIC ACCESSCONTROL IN THE CLOUD**” is done by me under the guidance of: **Mrs.M.D.ANTO PRAVEENA,M.C.A.,M.E.,(Ph.D.)**., Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE:CHENNAI

SIGNATURE OF THECANDIDATE

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, **Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the 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 **Mrs.M.D.ANTO PRAVEENA,M.C.A.,M.E.,(Ph.D.)**, **Associate Professor**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my projectwork

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

Enabling cryptographically enforced access controls for data hosted in untrusted cloud is attractive for many users and organizations. However, designing efficient cryptographically enforced dynamic access control system in the cloud is still challenging. In this paper, we propose Crypt-DAC, a system that provides practical cryptographic enforcement of dynamic access control. Crypt-DAC revokes access permissions by delegating the cloud to update encrypted data. In Crypt-DAC, a file is encrypted by asymmetric keylist which records a file key and a sequence of revocation keys. In each revocation, a dedicated administrator uploads a new revocation key to the cloud and requests it to encrypt the file with a new layer of encryption and update the encrypted key list accordingly. Crypt-DAC proposes three key techniques to constrain the size of key list and encryption layers. As a result, Crypt-DAC enforces dynamic access control that provides efficiency, as it does not require expensive decryption/encryption and uploading/re-uploading of large data at the administrator's side, and security, as it immediately revokes access permissions. We use formalization framework and system implementation to demonstrate the security and efficiency of our construction.

TABLE OF CONTENTS

ABSTRACT LIST OF FIGURES

V
VIII

CHAPTER No.	TITLE	PAGE No
1.	INTRODUCTION	1
	1.1 AIM	1
	1.2 SYNOPSIS	1
2.	SYSTEM ANALYSIS	2
	2.1 EXISTING SYSTEM	2
	2.1.1 PROBLEM STATEMENT	2
	2.2 PROPOSED SYSTEM	2
3.	REQUIREMENT SPECIFICATION	3
	3.1 INTRODUCTION	3
	3.2 HARDWARE AND SOFTWARE SPECIFICATION	4
	3.2.1 HARDWARE REQUIREMENTS	4
	3.2.2 SOFTWARE REQUIREMENTS	4
	3.3 TECHNOLOGIES USED	4
	3.3.1 JAVA	4
	3.3.1.1 INTRODUCTION TO JAVA	4
	3.3.1.2 WORKING OF JAVA	6
	3.3.1.3 THE JAVA PROGRAMMING LANGUAGE	6
	3.3.1.4 THE JAVA PLATFORM	8
	3.3.2 APACHE TOMCAT SERVER	10
	3.3.3 ANDRIOD INTRODUCTION	11
	3.3.3.1 LINUX KERNEL	12
	3.3.3.2 LIBRARIES	13
4	PROJECT PURPOSE AND SCOPE	14
	4.1 PURPOSE	14
	4.2 PROJECT SCOPE	14
	4.3 PRODUCT PERSPECTIVE	14
	4.4 SYSTEM FEATURES	15
	4.5 DESIGN AND IMPLEMENTATION CONSTRAINTS	15
	4.5.1 CONSTRAINTS IN ANALYSIS	15
	4.5.2 CONSTRAINTS IN DESIGN	15
	4.5.3 CONSTRAINTS IN IMPLEMENTATION	16
	4.6 OTHER NONFUNCTIONAL REQUIREMENTS	16
	4.6.1 PERFORMANCE REQUIREMENTS	16

	4.6.2 SAFETY REQUIREMENTS	16
5.	SYSTEM ARCHITECTURE	18
	5.1 ARCHITECTURE DIAGRAM	18
	5.2 SEQUENCE DIAGRAM	19
	5.3 USE CASE DIAGRAM	19
	5.4 ACTIVITY DIAGRAM	21
	5.5 COLLABORATION DIAGRAM	22
	5.6 DATA FLOW DIAGRAM	22
	5.7 CLASS DIAGRAM	24
6.	SYSTEM DESIGN	25
	6.1 MODULES	26
7.	CODING AND TESTING	28
	7.1 CODING	28
	7.2 CODING STANDARDS	28
	7.2.1 NAMING CONVENTIONS	29
	7.2.2 VALUE CONVENTIONS	29
	7.2.3 SCRIPT WRITING AND COMMENTING	29
	STANDARD	
	7.2.4 MESSAGE BOX FORMAT	30
	7.3 TEST PROCEDURE	30
	7.3.1 SYSTEM TESTING	30
	7.4 TEST DATA AND OUTPUT	31
	7.4.1 UNIT TESTING	31
	7.4.2 FUNCTIONAL TESTS	31
	7.4.3 PERFORMANCE TEST	31
	7.4.4 STRESS TEST	31
	7.4.5 STRUCTURED TEST	31
	7.4.6 INTEGRATION TESTING	32
	7.5 TESTING TECHNIQUES	33
	7.5.1 TESTING	33
	7.5.1.1 WHITE BOX TESTING	33
	7.5.1.2 BLACK BOX TESTING	34
	7.5.2 SOFTWARE TESTING STRATEGIES	34
	7.5.2.1 INTEGRATION TESTING	35
	7.5.2.2 PROGRAM TESTING	35
	7.5.2.3 SECURITY TESTING	35
	7.5.2.4 VALIDATION TESTING	36
	7.5.2.5 USER ACCEPTANCE TESTING	36
	CONCLUSION	36
	SOURCE CODE	37
	SCREENSHOTS	52

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.1	JAVA BYTE CODE	7
3.2	JAVA VIRTUAL MACHINE	8
3.3	JAVA PLATFORM	9
3.4	ANDRIOD APPLICATION	12
5.1	ARCHITECTURE DIAGRAM	18
5.2	SEQUENCE DIAGRAM	19
5.3	USECASE DIAGRAM	20
5.4	ACTIVITY DIAGRAM	21
5.5	COLLABORATION DIAGRAM	22
5.6	LEVEL 0	22
5.7	LEVEL 1	23
5.8	LEVEL 2	23
5.9	LEVEL 3	24
5.10	CLASS DIAGRAM	24

CHAPTER 1

INTRODUCTION

1.1 Aim:

The main aim of this project is to provide integrity of an organization's 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. 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. Crypt-DAC enforces dynamic access control that provides efficiency, as it does not require expensive decryption, re-encryption and uploading/re-uploading of large data at the administrator side, and security, as it immediately revokes access permissions.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

In existing system, the considerable advancements in cloud computing, users and organizations are finding it increasingly appealing to store and share data through cloud services. Cloud service providers (such as Amazon, Microsoft, Apple, etc.) provide abundant cloud-based services, ranging from small-scale personal services to large-scale industrial services. However, recent data breaches, such as releases of private photos, have raised concerns regarding the privacy of cloud-managed data. Actually, a cloud service provider is usually not secure due to design drawbacks of software and system vulnerability. Then the crypt-DAC proposes three key techniques. The administrator appends a new revocation key at the end of its key list and requests the cloud to update this key list in the policy data. The size of the key list however increases with the revocation operations, and a user has to download and decrypt a large key list in each file access. This method is called onion encryption.

2.1.1 Problem statement

Crypt-DAC proposes an adjustable onion encryption strategy to delegate the cloud to update file data. For a file, the administrator requests the cloud to encrypt the file with a new layer of encryption.

2.2 PROPOSED SYSTEM

To overcome these problems, we present Crypt-DAC, a cryptographically enforced dynamic access control system on un-trusted cloud. To overcome the onion encryption, we propose Tuple for security purpose. Every time users should upload the tuple file while accessing the cloud files. If the tuple verification is success you can access the files otherwise admin sent you a warning message three times and then admin will block you at the same time camera will capture your face and sent to admin.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 INTRODUCTION

With the considerable advancements in cloud computing, users and organizations are finding it increasingly appealing to store and share data through cloud services. Cloud service providers (such as Amazon, Microsoft, Apple, etc.) provide abundant cloud-based services, ranging from small-scale personal services to large-scale industrial services. However, recent data breaches, such as releases of private photos [10], have raised concerns regarding the privacy of cloud-managed data. Actually, a cloud service provider is usually not secure due to design drawbacks of software and system vulnerability [2], [3]. As such, a critical issue is how to enforce data access control on the potentially untrusted cloud. In response to these security issues, numerous works [1], [4]–[9] have been proposed to support access control on untrusted cloud services by leveraging cryptographic primitives. Advanced cryptographic primitives are applied for enforcing many access control paradigms. For example, attribute-based encryption (ABE) [5] is a cryptographic counterpart of attribute-based access control (ABAC) model [11]. However, previous works mainly consider static scenarios in which control policies rarely change. The previous works incur high overhead when access control policies need to be changed in practice. At a first glance, the revocation of a user's access can be done by revoking his access to the keys which the files are encrypted. This solution, however, is not secure as the user can keep a local copy of the keys before the revocation. To prevent such a problem, files have to be re-encrypted with new keys. This requires the file owner to download the file, re-encrypt the file, and upload it back for the cloud to update the previous encrypted file, incurring prohibitive communication overhead at the file owner's side.

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 1.8
- J2EE
- Tomcat 7.0
- MySQL

3.3 TECHNOLOGIES USED

- JAVA
- J2EE (JSTL, Spring boot), 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.

3.3.1.1 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 NC SA. 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.

3.3.1.2 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 (called attributes) and functions (called methods).

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, we execute 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 Windows platform, it will write the string passed to it to the user's program.

Java consists of two things:

- Programming language
- Platform

3.3.1.3 THE JAVA PROGRAMMING LANGUAGE

Java is a high-level programming language that is all of the following:

- Simple
- Object-oriented
- Distributed
- Interpreted
- Robust
- Secure

- Architecture-neutral
- Portable
- High-performance
- Multithreaded
- Dynamic

The code can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-mentioned objectives are as follows:

Java is unusual in that each Java program is both compiled and interpreted. With a compiler, you translate a Java program into an intermediate language called **Java byte codes** – the platform independent codes interpreted by the Java interpreter. With an interpreter, each Java byte code instruction is parsed and run on the computer. Compilation happens just once; interpretation occurs each time the program is executed. This figure illustrates how it works:

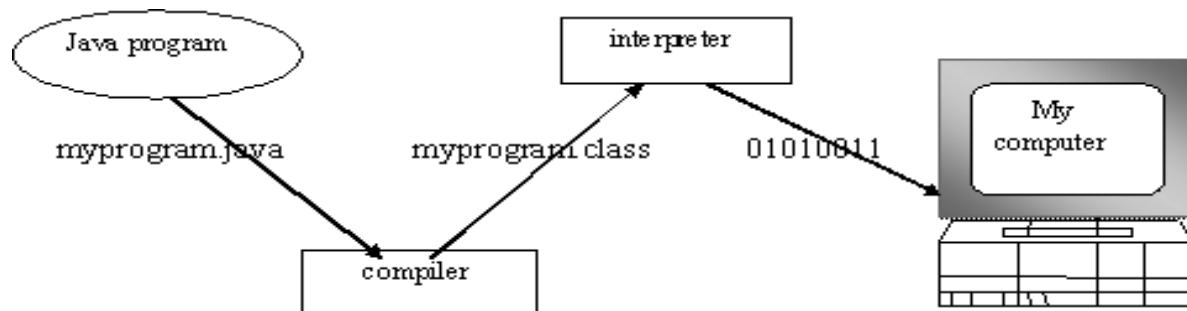


Fig.3.1 JAVA BYTE CODE

You can think of Java byte codes as the machine code instructions for the **Java Virtual Machine (JVM)**. Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of JVM. That JVM can also be implemented in hardware. Java byte codes help make "write once, run anywhere" possible.

You can compile your Java program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the JVM. For example, that same Java program can be run on Windows NT, Solaris and Macintosh.

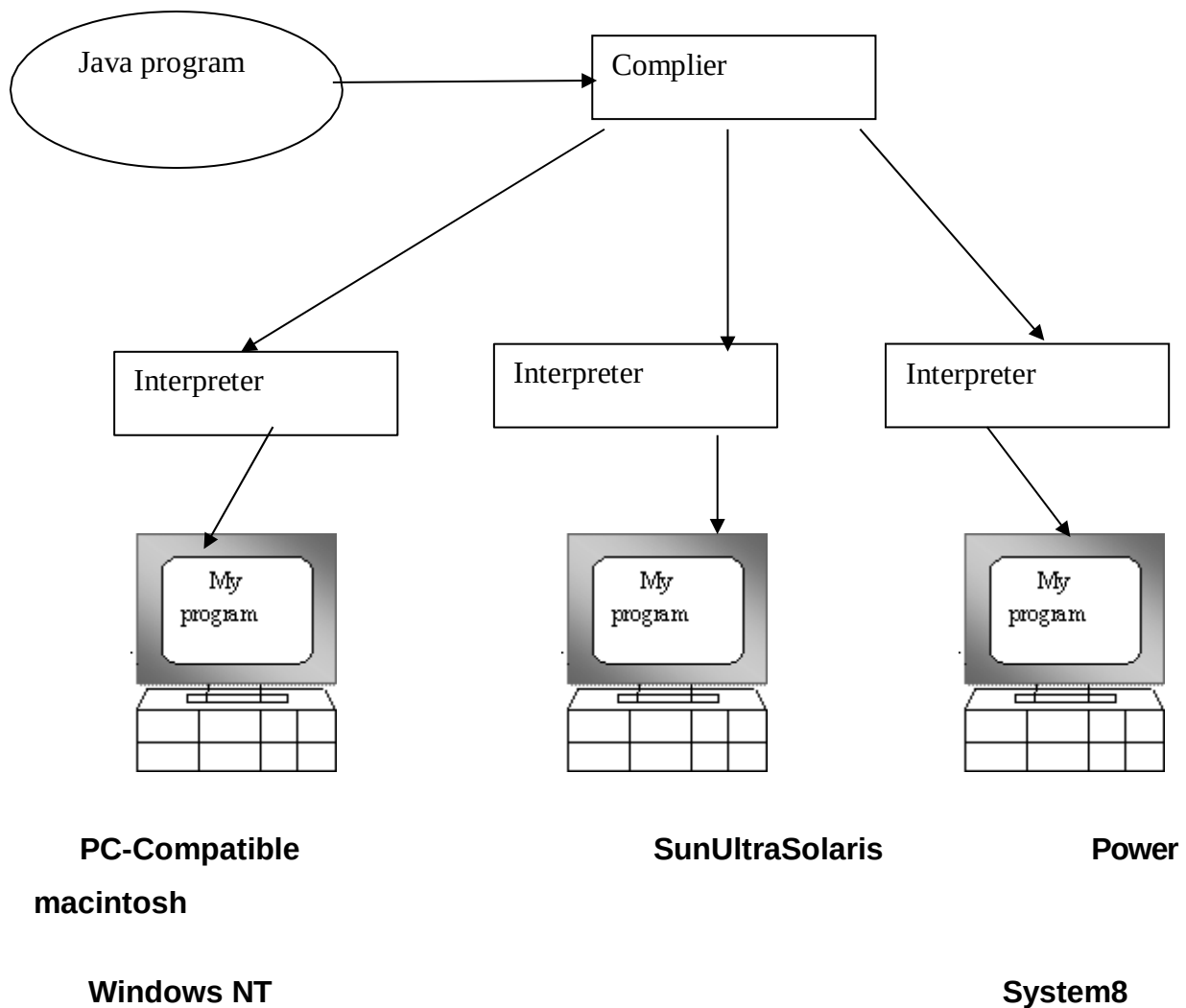


FIG 3.2 JAVA VIRTUAL MACHINE

3.3.1.4 THE JAVAPLATFORM

A platform is the hardware or software environment in which a program runs. 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. Most other platforms are described as a combination of hardware and operating system.

The Java platform has two components:

- The Java Virtual Machine(JVM)
- The Java Application Programming Interface (JavaAPI)

You've already been introduced to the JVM. 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 (**packages**) of related components. The following figure depicts a Java program, such as an application or applet, that's running on the Java platform. As the figure shows, the Java API and Virtual Machine insulates the Java program from hardware dependencies.

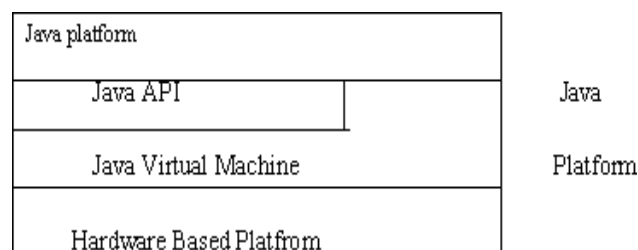


Fig.3.3 JAVA PLATFORM

As a platform-independent environment, Java can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte compilers can bring Java's performance close to that of native code without threatening portability.

3.3.2 APACHE TOMCATSERVER

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 Java Server 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.

Environment

Tomcat is a web server that supports servlets and JSPs. Tomcat comes with the Jasper compiler that compiles JSPs into servlets.

The Tomcat servlet engine is often used in combination with an Apache web server or other web servers. Tomcat can also function as an independent web server. Earlier in its development, the perception existed that standalone Tomcat was only suitable for development environments and other environments with minimal requirements for speed and transaction handling. However, that perception no longer exists; Tomcat is increasingly used as a standalone web server in high-traffic, high-availability environments.

Since its developers wrote Tomcat in Java, it runs on any operating system that has a JVM.

Product features

Tomcat 3.x (initial release)

- implements the Servlet 2.2 and JSP 1.1 specifications
- servlet reloading
- basic HTTP functionality Tomcat 4.x
- implements the Servlet 2.3 and JSP 1.2 specifications
- servlet container redesigned as Catalina

- JSP engine redesigned as Jasper
- Coyote connector
- Java Management Extensions (JMX), JSP and Struts-based administration
- Tomcat 5.x
- implements the Servlet 2.4 and JSP 2.0 specifications
- reduced garbage collection, improved performance and scalability
- native Windows and Unix wrappers for platform integration
- faster JSP parsing

History

Tomcat started off as a servlet specification implementation by James Duncan Davidson, a software architect at Sun. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation.

Davidson had initially hoped that the project would become open-sourced and, since most open-source projects had O'Reilly books associated with them featuring an animal on the cover, he wanted to name the project after an animal. He came up with Tomcat since he reasoned the animal represented something that could take care of and fend for itself. His wish to see an animal cover eventually came true when O'Reilly published their Tomcat book with a tomcat on the cover.

3.3.3 Android Introduction:

Android is a Linux-based operating system it is designed primarily for touchscreen mobile devices such as smart phones and tablet computers. The operating system has developed a lot in last 15 years starting from black and white phones to recent smart phones or mini computers. One of the most widely used mobile OS these days is android. The android is software that was founded in Palo Alto of California in 2003.

The android is a powerful operating system and it supports large number of applications in Smart phones. These applications are more comfortable and advanced for the users. The hardware that supports android software is based on ARM architecture platform. The android is an open-source operating system means that it's free and any one can use it. The android has got millions of apps available that can help you managing

your life one or other way and it is available low cost in market at that reasons android is very popular.

The android development supports with the full java programming language. Even other packages that are API and JSE are not supported. The first version 1.0 of android development kit (SDK) was released in 2008 and latest updated version is jellybean.

The android is an operating system and is a stack of software components which is divided into five sections and four main layers that is



FIG 3.4 ANDRIOD APPLICATION

3.3.3.1 Linux kernel:

The android uses the powerful Linux kernel and it supports wider range of hardware drivers. The kernel is the heart of the operating system that manages input and output

requests from software. This provides basic system functionalities like process management, memory management, device management like camera, keypad, display etc the kernel handles all the things. The Linux is really good at networking and it is not necessary to interface it to the peripheral hardware. The kernel itself does not interact directly with the user but rather interacts with the shell and other programs as well as with the hardware devices on the system.

3.3.3.2 Libraries:

On top of a Linux kernel there is a set of libraries including open-source web browsers such as WebKit, library libc. These libraries are used to play and record audio and video. The SQLite is a database which is useful for storage and sharing of application data. The SSL libraries are responsible for internet security etc.

CHAPTER 4

Project Purpose and Scope

4.1 Purpose

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

4.2 Project Scope

To overcome these problems, we present Crypt-DAC, a cryptographically enforced dynamic access control system on un-trusted cloud. To overcome the onion encryption, we propose Tuple for security purpose. Every time users should upload the tuple file while accessing the cloud files. If the tuple verification is successful you can access the files otherwise admin sent you a warning message three times and then admin will block you at the same time camera will capture your face and sent to admin.

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. 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. Crypt-DAC enforces dynamic access control that provides efficiency, as it does not require expensive decryption, re-encryption and uploading/re-uploading of large data at the administrator's side, and security, as it

immediately revokes access permissions.

4.4 SystemFeatures

We present Crypt-DAC, a cryptographically enforced dynamic access control system on un-trusted cloud. Crypt-DAC delegates the cloud to update encrypted files in permission revocations. In Crypt-DAC, a file is encrypted by a symmetric key list which records a file key and a sequence of revocation keys. In a revocation, the administrator uploads a new revocation key to the cloud, which encrypts the file with a new layer of encryption and updates the encrypted key list accordingly. Same as previous works [12], [23], we assume a honest-but-curious cloud, i.e., the cloud is honest to perform the required commends (such as re-encryption of files and properly update previous encrypted files) but is curious to passively gathering sensitive information. Although the basicideaoflayeredencryptionissimple,itentailstremendoustechnicalchallenges.For instance, the size of key list and encryption layers would increase as the number of revocation operations, which incurs additional decryption overhead for users to access files.Toovercomesuchaproblem,Crypt-DACproposethreekeytechniquesasfollows.

4.5 Design and ImplementationConstraints

4.5.1 Constraints inAnalysis

- ◆ Constraints as InformalText
- ◆ Constraints as OperationalRestrictions
- ◆ Constraints Integrated in Existing ModelConcepts
- ◆ Constraints as a SeparateConcept
- ◆ Constraints Implied by the ModelStructure

4.5.2 Constraints inDesign

- ◆ Determination of the InvolvedClasses
- ◆ Determination of the InvolvedObjects
- ◆ Determination of the InvolvedActions
- ◆ Determination of the RequireClauses
- ◆ Global actions and ConstraintRealization

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 Other Nonfunctional Requirements

4.6.1 Performance Requirements

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the webservice;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

4.6.2 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

SYSTEM ARCHITECTURE

5.1 ArchitectureDiagram:

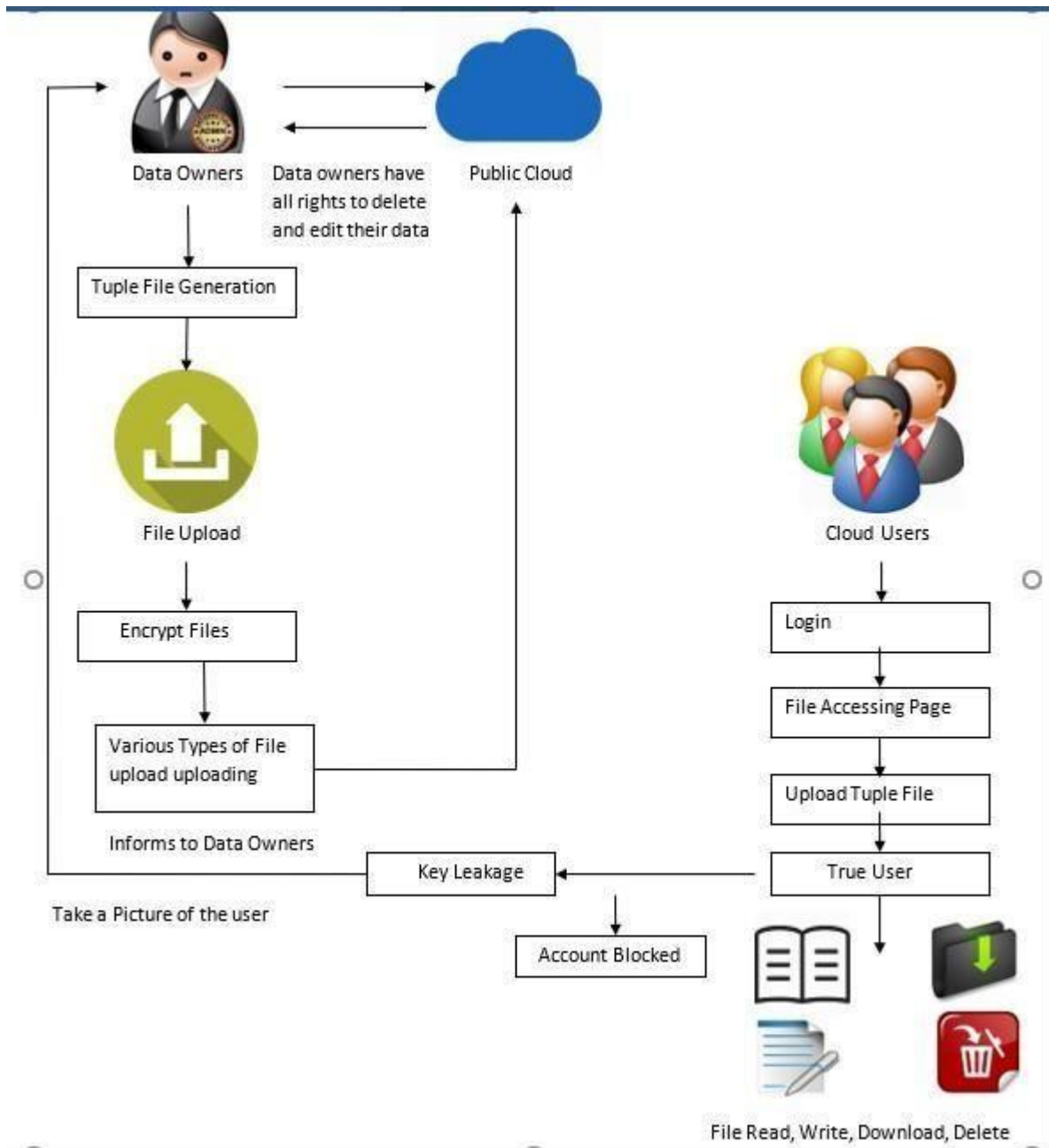


Fig: 5.1 ARCHITECTURE DIAGRAM

5.2 SequenceDiagram:

A Sequencediagramisakindofinteractiondiagramthatshowshowprocessesoperate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timingdiagram.

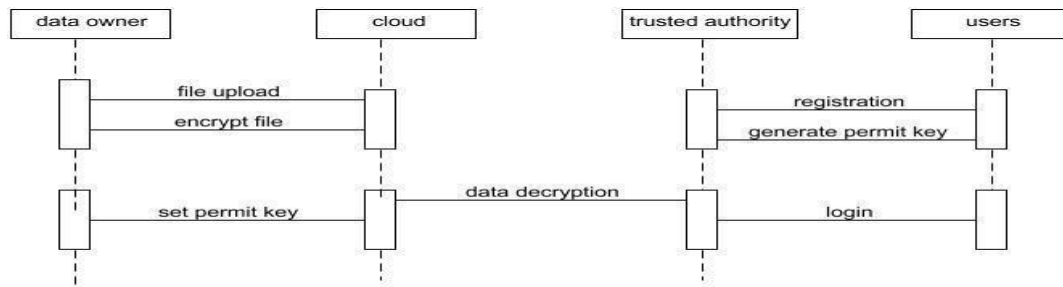


FIG 5.2 Sequence Diagram

5.3 Use CaseDiagram:

Unified Modeling Language (UML) is a standardized general-purpose modeling languageinthe fieldofsoftwareengineering.The standardismanagedandwascreated 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 underdevelopment.

5.3.1. USECASEDIAGRAM

A Use case Diagram is used to present a graphical overview of the functionality providedbyasystemintermsofactors,theirgoalsandanydependenciesbetweenthose usecases.

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.

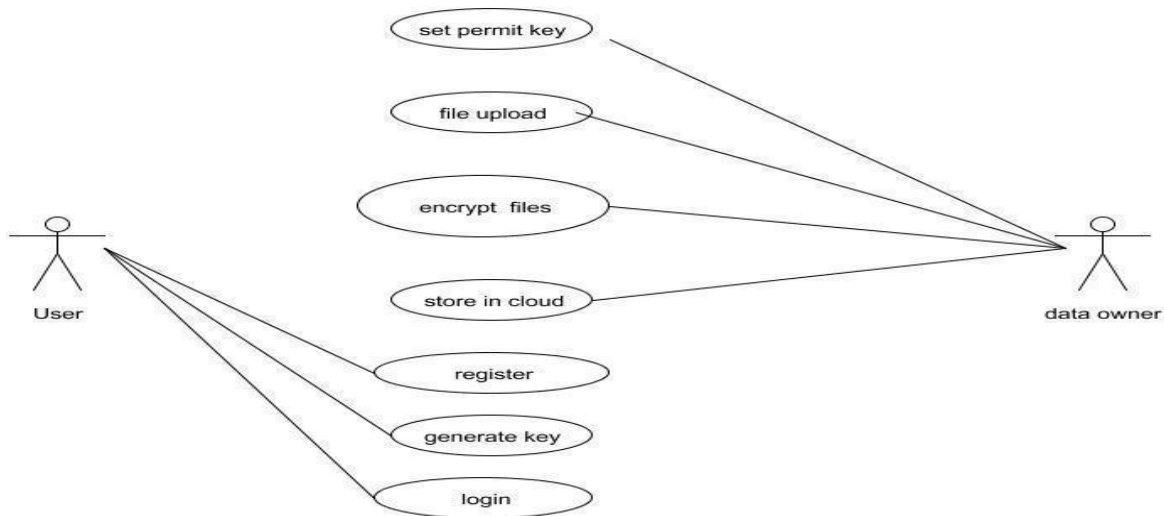


FIG 5.3 USECASE DIAGRAM

5.4 ActivityDiagram:

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.

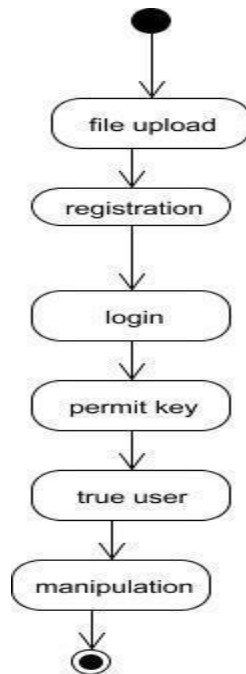


FIG 5.4 Activity Diagram

5.5 CollaborationDiagram:

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.

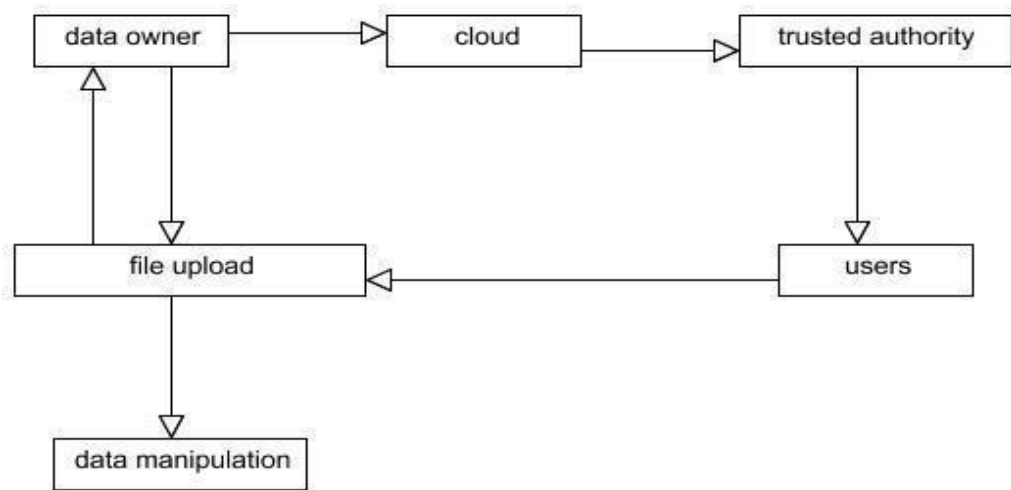


FIG 5.5 Collaboration Diagram

5.6 DATA FLOWDIAGRAM:

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 0

Level 1:

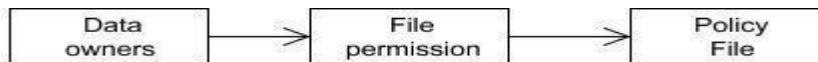


FIG 5.7 LEVEL 1

Level 2:



FIG 5.8 LEVEL 2

Level 3:

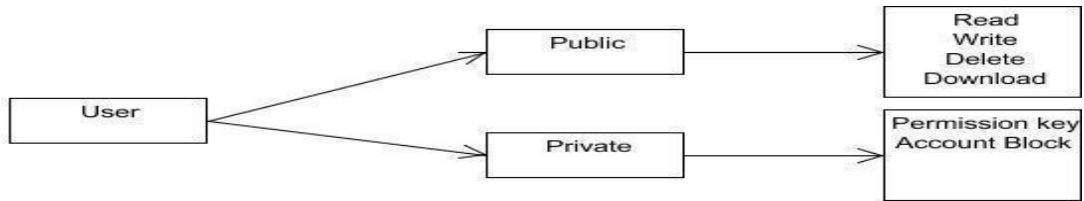


FIG 5.9 LEVEL 3

5.7 ClassDiagram

A Class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

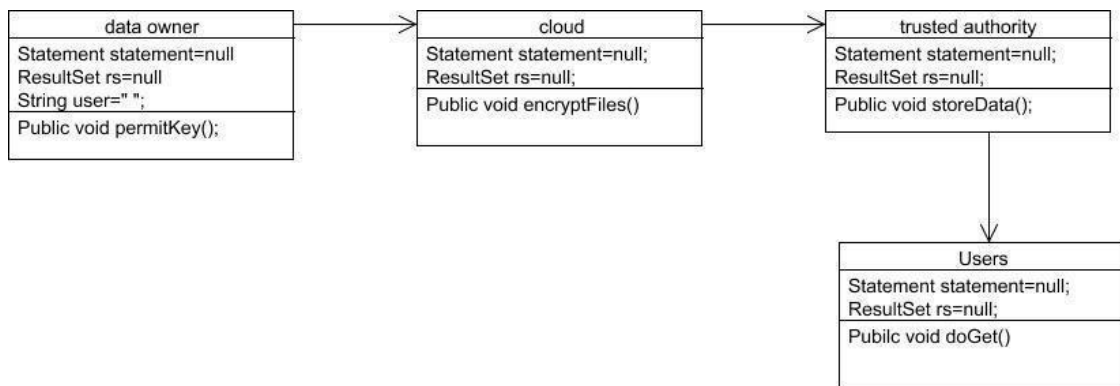


FIG 5.10 CLASS DIAGRAM

CHAPTER 6

SYSTEM DESIGN

The roles will be created for employee and the cloud authority. The roles will be created based on their designation. The employee and the cloud authority will get added based on their designation and roles. The admin will upload the file which is of two types. They are public and private files. The admin will add the news. If the file is public, it does not contain any access permission. If the file is private, then the tuples will get generated. Here file permission keys are issued to the employees in the organization based on their experience and position to their registered. Senior Employees have all the permission to access the files (read, write, delete, & download). Freshers or trainee 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 deleting the data. 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. Tuples are the encrypted PDF files which will be generated while the employee logs in. These tuples will get generated based on the roles of employee and the cloud authority. Authorized DUs are able to access (e.g., read, write, download, delete and decrypt) the outsourced data. While entering the password for re-encryption system, it will generate a attributeset for their role in background validate that the user has all rights to access the data. If the attributeset 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. If any employee does an illegal access of files without any permission, they will be warned for 3 times. If they continue the access, they will get captured by the camera and send as a notification to the admin.

6.1 MODULES

- **RoleCreation**
- **Admin File Upload**
- **TupleGeneration**
- **User File Access**

1. RoleCreation

The roles will be created for employee and the cloud authority. The roles will be created based on their designation. The employee and the cloud authority will get added based on their designation and roles.

2. Admin FileUpload

The admin will upload the file which is of two types. They are public and private files. The admin will add the news. If the file is public, it does not contain any access permission. If the file is private, then the tuples will get generated.

3. Tuple Generation

Here file permission keys are issued to the employees in the organization based on their experience and position to their registered. Senior Employees have all the permission to access the files (read, write, delete, & download). Freshers or trainee 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 deleting the data. 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. Tuples are the encrypted PDF files which will be generated while the employee logs in. These tuples will get generated based on the roles of employee and the cloud authority.

4. User File Access

Authorized DUs are able to access (e.g., read, write, download, delete and decrypt) the outsourced data. While entering the password for re-encryption system, it will generate attribute set for their role in background validate that the user has all rights to access the data. If the attribute set is not matched to the Data Owner's policy files they will be claimed as guilty. If we ask them, we will find who leaked the key to the junior employees. If any employee does an illegal access of files without any permission, they will be warned for 3 times. If they continue the access, they will get captured by the camera and send as a notification to the admin.

CHAPTER 7

CODING AND TESTING

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 easy 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 VALUECONVENTIONS

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 BOXFORMAT

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.

7.3 TESTPROCEDURE

7.3.1 SYSTEMTESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

7.4 TEST DATA AND OUTPUT

7.4.1 UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

7.4.2 FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

Three types of tests in Functional test:

- Performance Test
- Stress Test
- Structure Test

7.4.3 PERFORMANCE TEST

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

7.4.4 STRESS TEST

Stress Test is those tests designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

7.4.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information.

7.4.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And soon. The advantages of this approach are that, interfaced dispenses can be easily found and corrected.

The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

7.5 TESTING TECHNIQUES / TESTING STRATEGIES

7.5.1 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 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 has a high probability of finding any yet undiscovered error. A successful test is one that uncovers an as-yet-undiscovered error. Any engineering product can be tested in one of the two ways:

7.5.1.1 WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation

- Cyclometric complexity
- Deriving testcases
- Graph matricesControl

7.5.1.2 BLACK BOXTESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testingmethods
- Equivalencepartitioning
- Boundary value analysis
- Comparisontesting

7.5.2 SOFTWARE TESTINGSTRATEGIES:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason, a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works “outward” toward the integration of the entire computer-basedsystem.
- Different testing techniques are appropriate at different points intime.
- The developer of the software and an independent test group conducts testing.
- Testing and Debugging are different activities but debugging must be accommodated in any testingstrategy.

7.5.2.1 INTEGRATION TESTING:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

7.5.2.2 PROGRAM TESTING:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax errors. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-of-range items and invalid combinations. Since the compiler will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses, a relational operator or on arithmetic expression. Condition testing method focuses on testing each condition in the program the purpose of condition test is to deduct not only errors in the condition of a program but also other errors in the program.

7.5.2.3 SECURITY TESTING:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

7.5.2.4 VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- * The function or performance characteristics confirm to specifications and are accepted.
- * A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic

7.5.2.5 USER ACCEPTANCE TESTING

User acceptance of the system is a key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regard to the following points.

- Input screen design.
- Output screen design.

Conclusion:

Thus, to provide integrity of an organization data which is in public cloud is achieved successfully.

SOURCE CODE

Wrapper properties

distributionUrl=https://repo.maven.apache.org/maven2/org/apache/maven/apache-maven/3.6.2/apache-maven-3.6.2-bin.zip

wrapperUrl=https://repo.maven.apache.org/maven2/io/takari/maven-wrapper/0.5.5/maven-wrapper-0.5.5.jar

Wrapper Downloader

/*

* Copyright 2012-2019 the original author or authors.

*

* Licensed under the Apache License, Version 2.0 (the "License");

* you may not use this file except in compliance with the License.

* You may obtain a copy of the License at

*

* <https://www.apache.org/licenses/LICENSE-2.0>

*

* Unless required by applicable law or agreed to in writing, software

* distributed under the License is distributed on an "AS IS" BASIS,

* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

* See the License for the specific language governing permissions and

* limitations under theLicense.

*/

import java.net.*;

import java.io.*;

import java.nio.channels.*;

import java.util.Properties;

public class MavenWrapperDownloader {

private static final String WRAPPER_VERSION = "0.5.5";

/**

* Default URL to download the maven-wrapper.jar from, if no 'downloadUrl' is provided.

*/

private static final String DEFAULT_DOWNLOAD_URL =
"https://repo.maven.apache.org/maven2/io/takari/maven-wrapper/"

+ WRAPPER_VERSION + "/maven-wrapper-" + WRAPPER_VERSION + ".jar";

/**

* Path to the maven-wrapper.properties file, which might contain a downloadUrl property to

* use instead of the default one.

```

*/

private static final String MAVEN_WRAPPER_PROPERTIES_PATH =

    ".mvn/wrapper/maven-wrapper.properties";

/**

 * Path where the maven-wrapper.jar will be saved to.

 */

private static final String MAVEN_WRAPPER_JAR_PATH =

    ".mvn/wrapper/maven-wrapper.jar";

/**

 * Name of the property which should be used to override the default download url for
thewrapper.

 */

private static final String PROPERTY_NAME_WRAPPER_URL = "wrapperUrl";

public static void main(String args[]) {

    System.out.println("- Downloader started");

    File baseDirectory = new File(args[0]);

    System.out.println("- Using base directory: " + baseDirectory.getAbsolutePath());

```

```

// If the maven-wrapper.properties exists, read it and check if it contains a custom
// wrapperUrl parameter.

File mavenWrapperPropertyFile = new File(baseDirectory,
MAVEN_WRAPPER_PROPERTIES_PATH);

String url = DEFAULT_DOWNLOAD_URL;

if(mavenWrapperPropertyFile.exists()) {

    FileInputStream mavenWrapperPropertyFileInputStream = null;

    try {

        mavenWrapperPropertyFileInputStream = new
FileInputStream(mavenWrapperPropertyFile);

        Properties mavenWrapperProperties = new Properties();

        mavenWrapperProperties.load(mavenWrapperPropertyFileInputStream);

        url =
mavenWrapperProperties.getProperty(PROPERTY_NAME_WRAPPER_URL, url);

    } catch (IOException e) {

        System.out.println("- ERROR loading '" +
MAVEN_WRAPPER_PROPERTIES_PATH + "'");

    } finally {

        try {

            if(mavenWrapperPropertyFileInputStream != null) {

                mavenWrapperPropertyFileInputStream.close();

            }

        }

```

```

        } catch (IOException e) {

            // Ignore ...

        }

    }

}

System.out.println("- Downloading from: " + url);


File outputFile = new File(baseDirectory.getAbsolutePath(),
MAVEN_WRAPPER_JAR_PATH);

if(!outputFile.getParentFile().exists()) {

    if(!outputFile.getParentFile().mkdirs()) {

        System.out.println(

            "- ERROR creating output directory '" +
outputFile.getParentFile().getAbsolutePath() + "'");

    }

}

System.out.println("- Downloading to: " + outputFile.getAbsolutePath());

try {

    downloadFileFromURL(url, outputFile);

    System.out.println("Done");

    System.exit(0);

```

```

    } catch (Throwable e){

        System.out.println("- Error downloading");

        e.printStackTrace();

        System.exit(1);

    }

}

```

private static void downloadFileFromURL(String urlString, File destination) throws Exception {

```

    if (System.getenv("MVNW_USERNAME") != null &&
        System.getenv("MVNW_PASSWORD") != null) {

        String username = System.getenv("MVNW_USERNAME");

        char[] password = System.getenv("MVNW_PASSWORD").toCharArray();

        Authenticator.setDefault(new Authenticator() {

            @Override

            protected PasswordAuthentication getPasswordAuthentication() {

                return new PasswordAuthentication(username, password);

            }

        });

    }

    URL website = new URL(urlString);

```

```

        ReadableByteChannel rbc;

        rbc = Channels.newChannel(website.openStream());

        FileOutputStream fos = new FileOutputStream(destination);

        fos.getChannel().transferFrom(rbc, 0, Long.MAX_VALUE);

        fos.close();

        rbc.close();

    }
}

```

Java Main

```

package com.cryptdac;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.builder.SpringApplicationBuilder;

import org.springframework.boot.web.servlet.support.SpringBootServletInitializer;

import org.springframework.context.annotation.ComponentScan;

import com.cryptdac.service.UserService;

@SpringBootApplication

```

```

@ComponentScan(basePackages = { "com.cryptdac.*" })

public class CryptdacApplication extends SpringBootServletInitializer {

    //

    //      @Autowired

    //      UserService userService;

    @Override

    protected SpringApplicationBuilder configure(SpringApplicationBuilder
application) {

        return application.sources(CryptdacApplication.class);

    }

    public static void main(String[] args) {

        SpringApplication.run(CryptdacApplication.class, args);

    }

    public CryptdacApplication() {

        super();

        setRegisterErrorPageFilter(false); // <- this one

    //      userService.addRole();

```

```
    }  
}
```

Java Application Testing

```
package com.cryptdac;
```

```
import org.junit.Test;
```

```
import org.junit.runner.RunWith;
```

```
import org.springframework.boot.test.context.SpringBootTest;
```

```
import org.springframework.test.context.junit4.SpringRunner;
```

```
@RunWith(SpringRunner.class)
```

```
@SpringBootTest
```

```
public class CryptdacApplicationTests {
```

```
    @Test
```

```
    public void contextLoads() {
```

```
    }
```

Classes

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

server.servlet.context-path=/cryptdac

#spring.main.banner-mode=off

#server.servlet-path = /*

spring.resources.add-mappings=true

create and drop tables and sequences, loadsimport.sql

spring.jpa.hibernate.ddl-auto=update

MySql settings

spring.datasource.url=jdbc:mysql://localhost:3306/crypt_dac

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver.class=com.mysql.jdbc.Driver

spring.jpa.properties.hibernate.show_sql=true

logging

#logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} %-5level %logger{36} - %msg%n

logging.level.org.hibernate.SQL=debug

#logging.level.org.hibernate.type=TRACE

#logging.level.org.hibernate.type.descriptor.sql=trace

#logging.level.=debug

#multipart

spring.http.multipart.max-file-size=10MB

spring.http.multipart.max-request-size=10MB

#CAPTCHA

google.recaptcha.secret=6Ledn9IUAAAAAG-kejvTcsw0VUjG7zSXgGI3BOx7

#FILE PATH

FILES_FOLDER_PATH=F:/CRYPTDAC_UPLOADS/

Mail Properties

#EMAIL PROPERTIES

MAIL_HOST_NAME="smtp.gmail.com"

MAIL_PORT_NUMBER="587";

MAIL_USERNAME="cryptdac@gmail.com"

MAIL_PASSWORD="cryptdac@1994"

MAIL_FORGOT_PASSWORD_SUBJECT="Password Recovery"

MAIL_FORGOT_PASSWORD_BODY="<html><body>Hi {0},
 Your Password For {1} is : {2} </body></html>"

MAIL_TUPLE_SUBJECT="TUPLE FILE PERMISSIONS"

MAIL_TUPLE_BODY="Dear {0}, Please find an attachment.
 To open the TUPLE please enter the first four letters of your username and your year of birth</body>"

Gitignore

HELP.md

target/

!.mvn/wrapper/maven-wrapper.jar

!**/src/main/**

!**/src/test/**

STS

.apt_generated

.classpath

.factorypath

.project

.settings

.springBeans

.sts4-cache

IntelliJ IDEA

.idea

*.iws

*.iml

*.ipr

NetBeans

/nbproject/private/

/nbbuild/

/dist/

/nbdist/

/.nb-gradle/

build/

VS Code

.vscode/

Project

<?xml version="1.0" encoding="UTF-8"?>

<projectDescription>

 <name>cryptdac</name>

 <comment></comment>

 <projects>

 </projects>

 <buildSpec>

 <buildCommand>

 <name>org.eclipse.wst.common.project.facet.core.builder</name>

```

        <arguments>

        </arguments>

    </buildCommand>

    <buildCommand>

        <name>org.eclipse.jdt.core.javabuilder</name>

        <arguments>

        </arguments>

    </buildCommand>

    <buildCommand>

        <name>org.springframework.ide.eclipse.core.springbuilder</name>

        <arguments>

        </arguments>

    </buildCommand>

    <buildCommand>

        <name>org.eclipse.wst.validation.validationbuilder</name>

        <arguments>

        </arguments>

    </buildCommand>

    <buildCommand>

        <name>org.eclipse.m2e.core.maven2Builder</name>

        <arguments>

```

```
        </arguments>

    </buildCommand>

</buildSpec>

<natures>

    <nature>org.eclipse.jem.workbench.JavaEMFNature</nature>

    <nature>org.eclipse.wst.common.modulecore.ModuleCoreNature</nature>

    <nature>org.springframework.ide.eclipse.core.springnature</nature>

    <nature>org.eclipse.jdt.core.javanature</nature>

    <nature>org.eclipse.m2e.core.maven2Nature</nature>

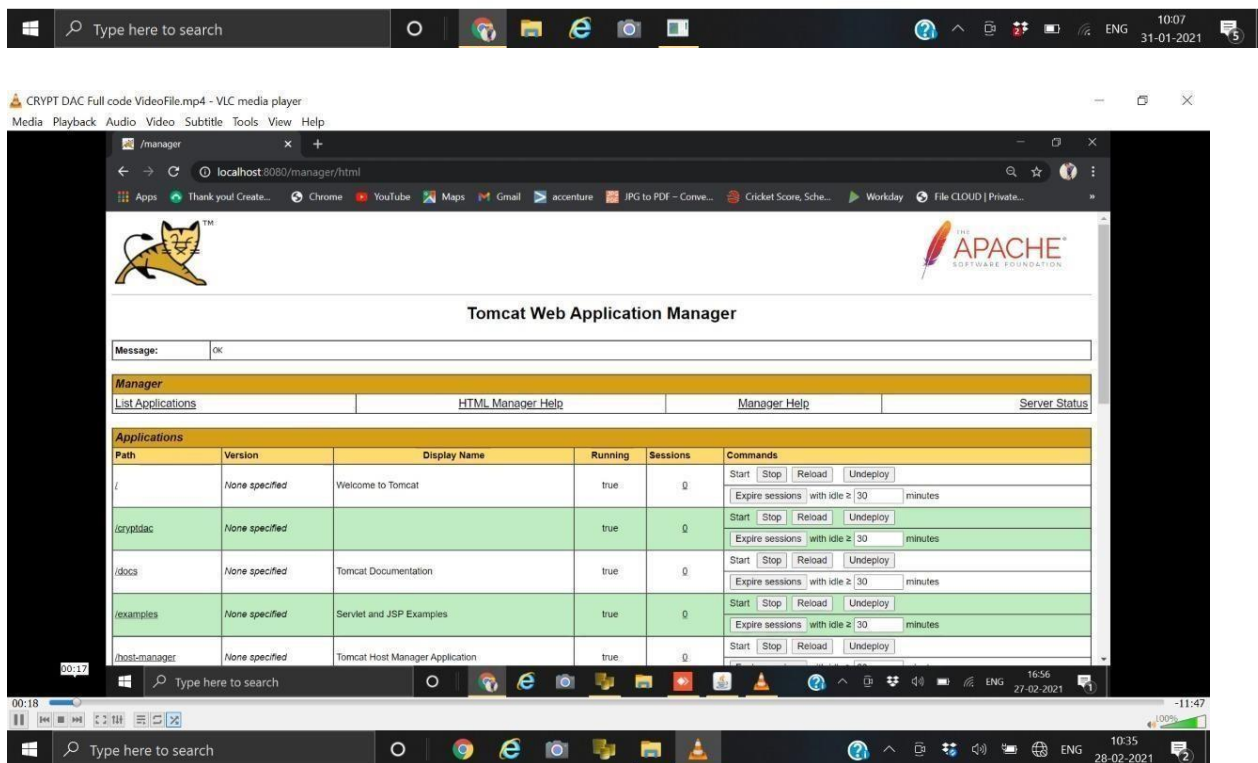
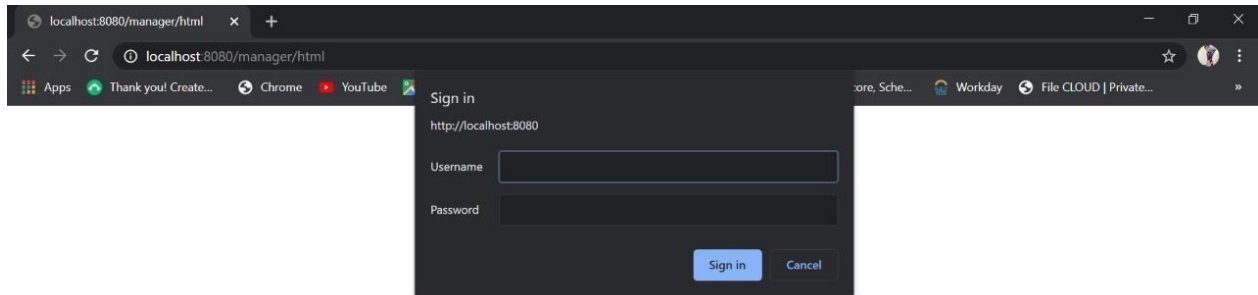
    <nature>org.eclipse.wst.common.project.facet.core.nature</nature>

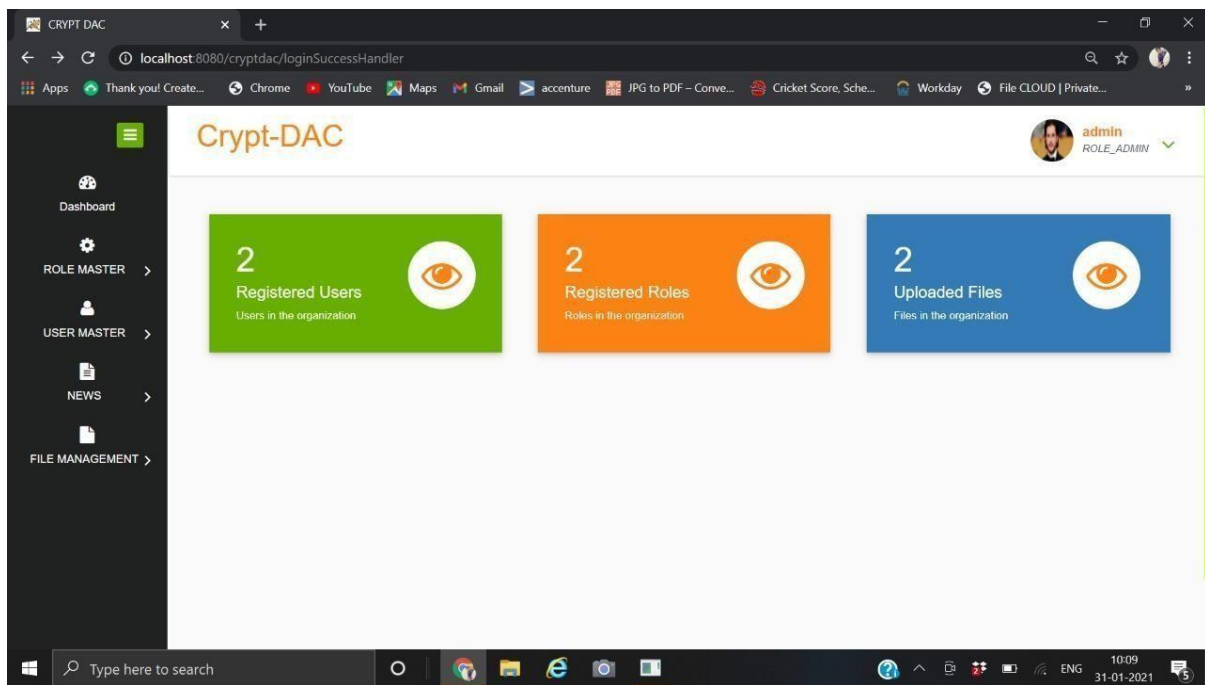
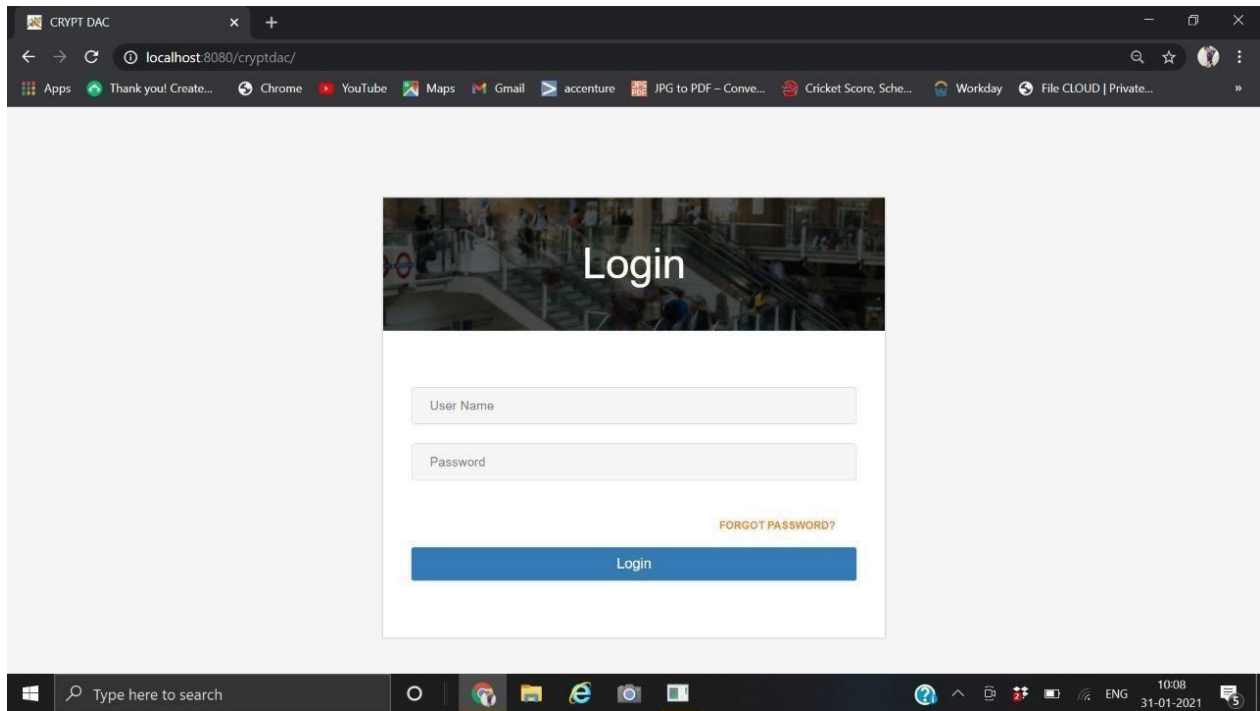
    <nature>org.eclipse.wst.jsdt.core.jsNature</nature>

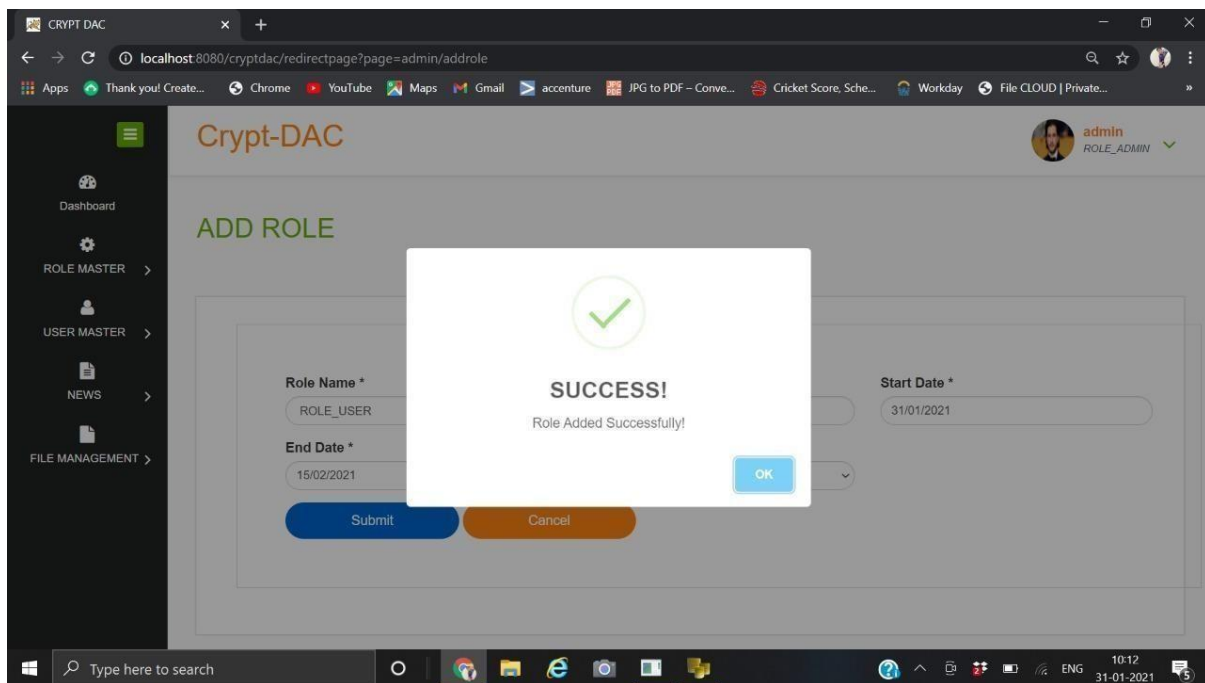
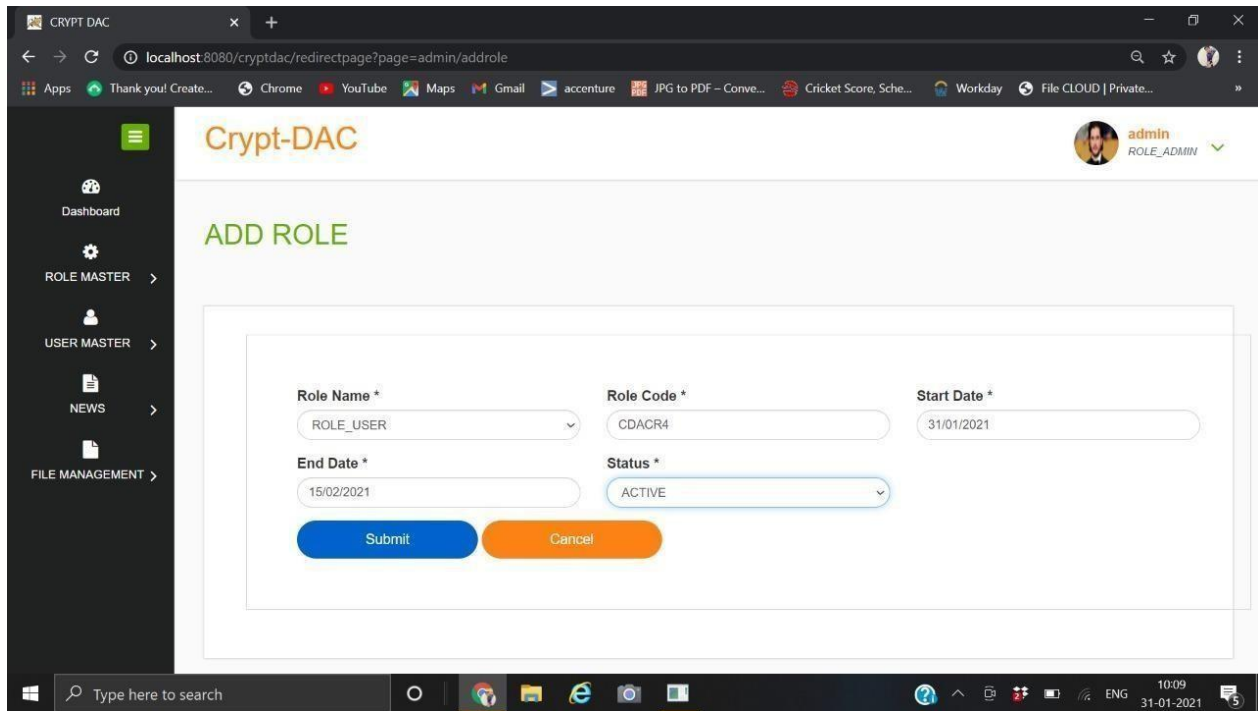
</natures>

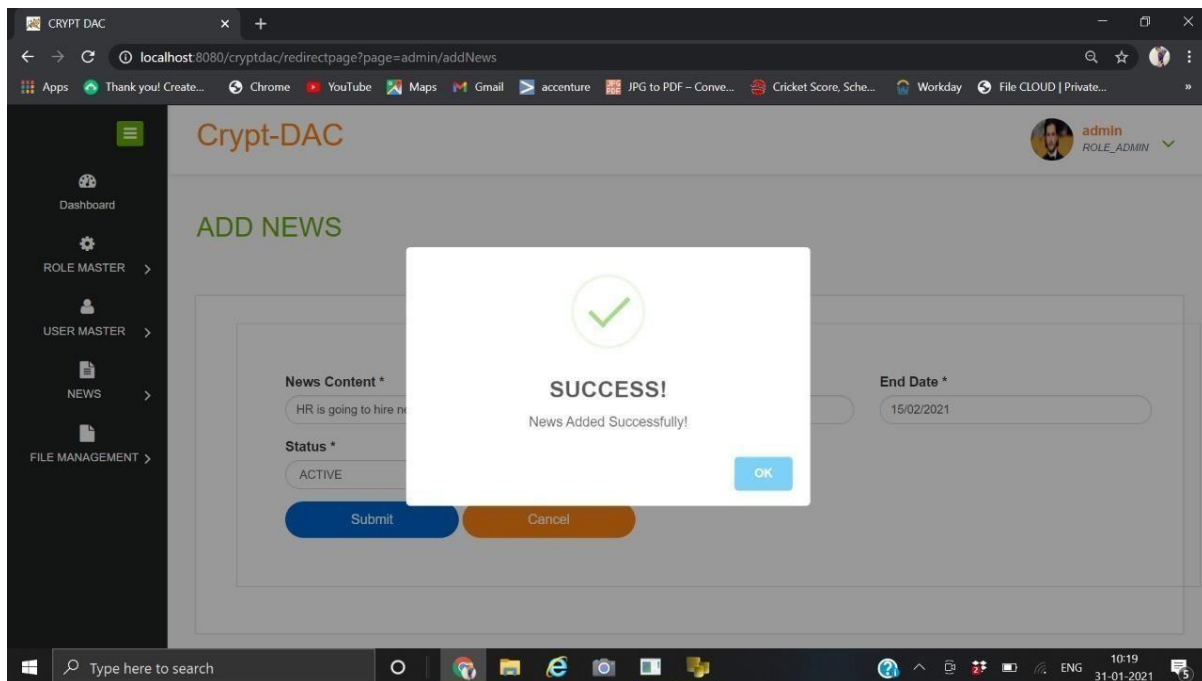
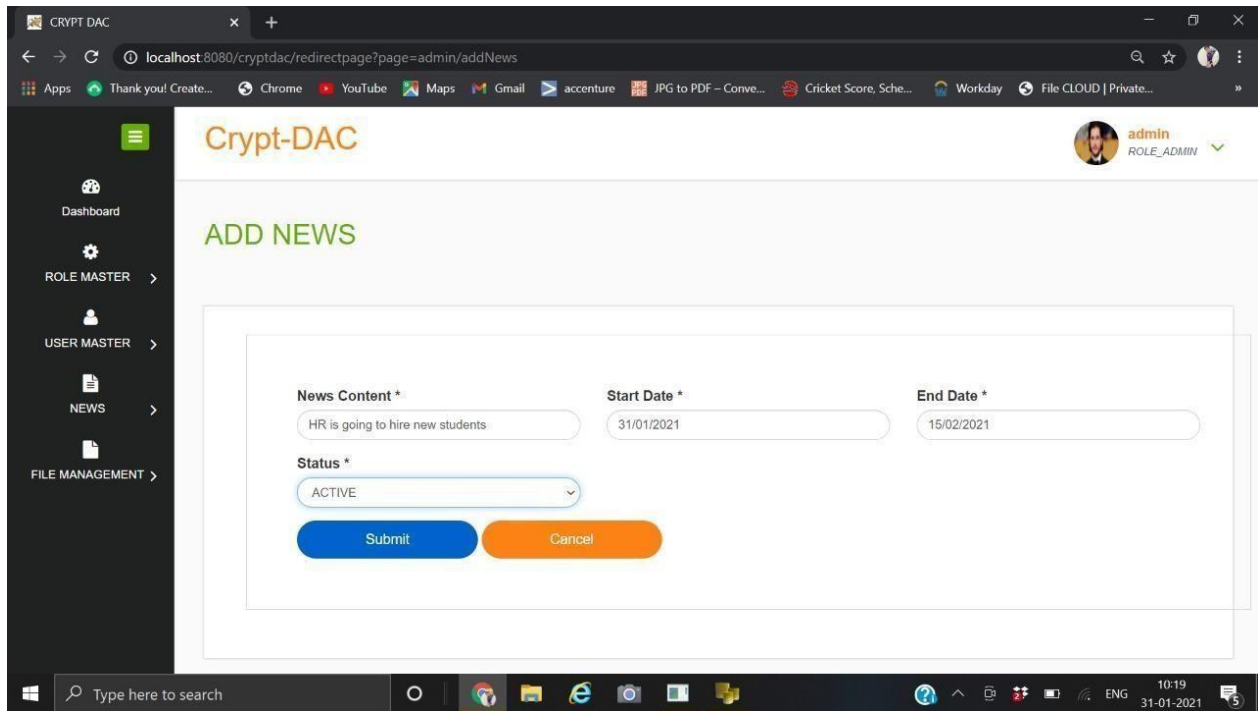
</projectDescription>
```

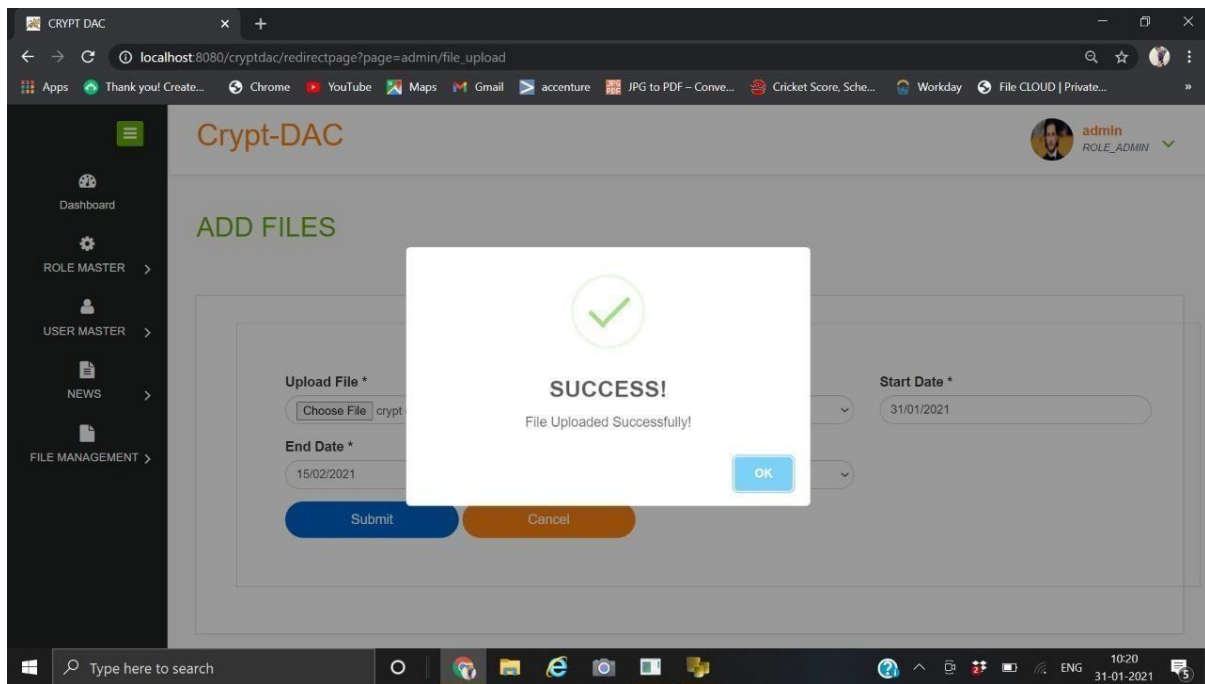
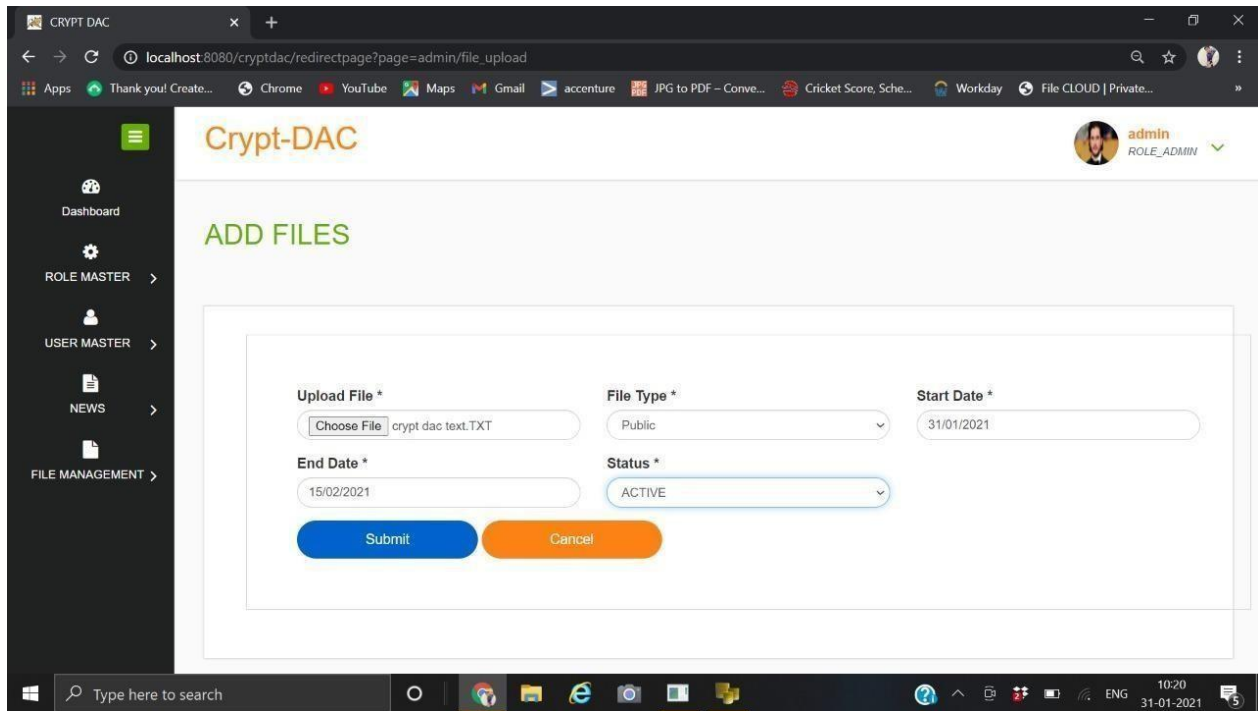
Screenshots

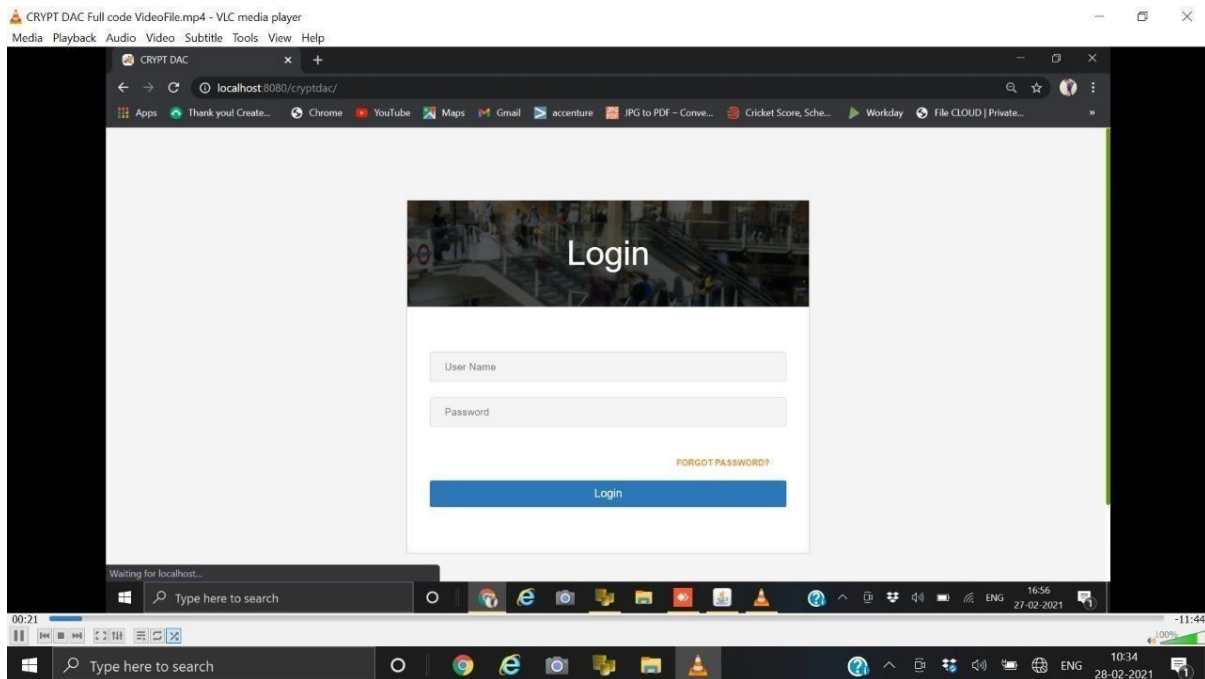
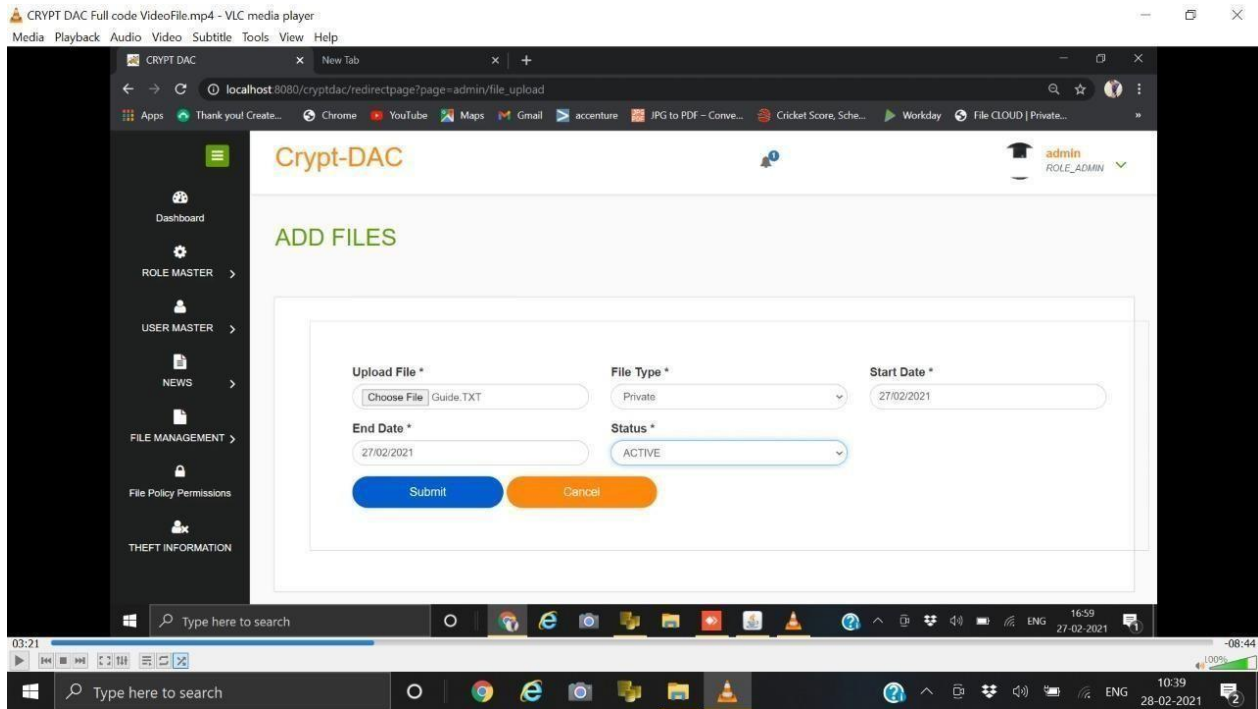


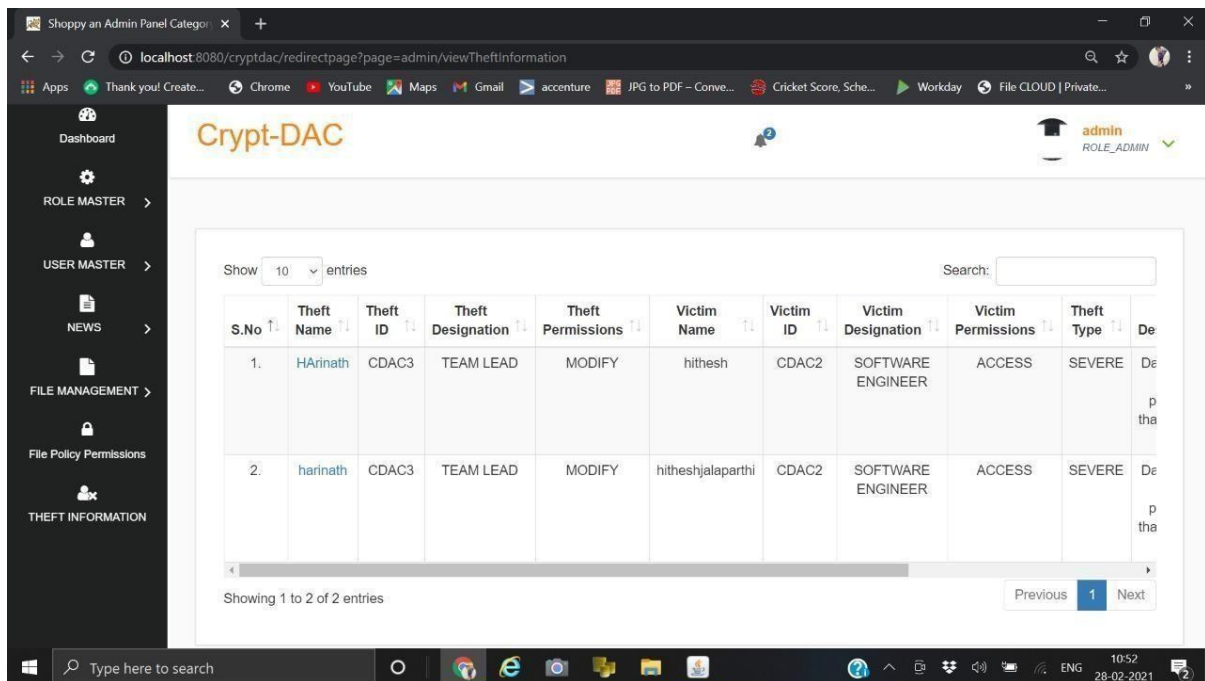
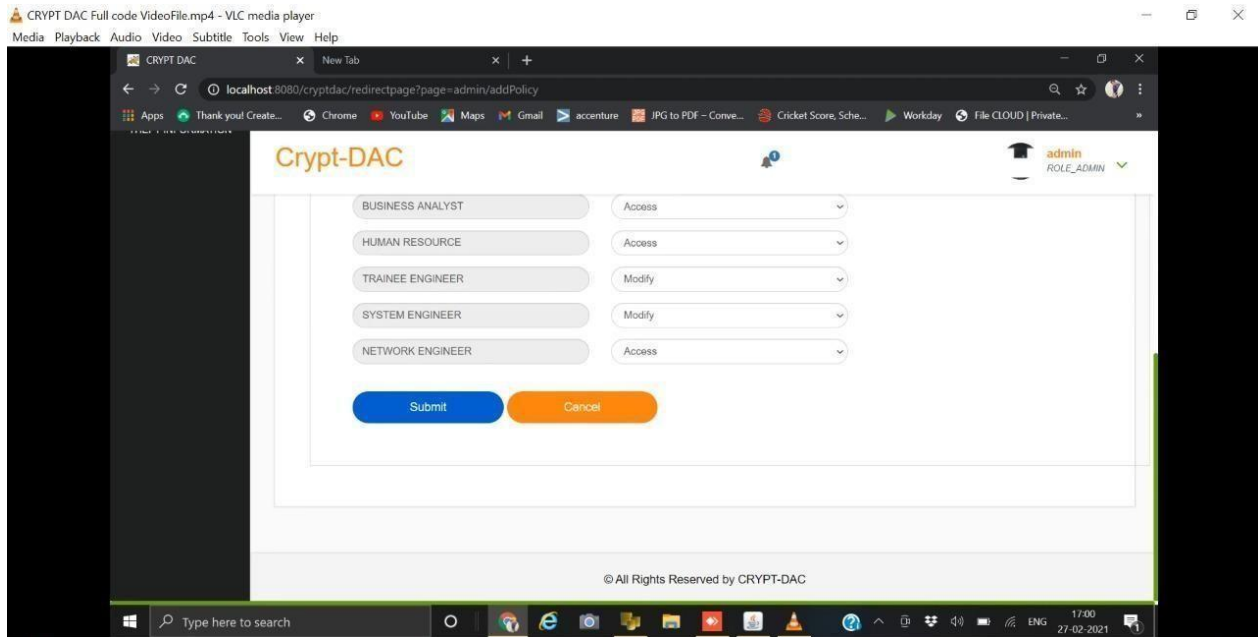












Shoppy an Admin Panel Category: x +

localhost:8080/cryptdac/redirectpage?page=admin/viewTheftInformation

Crypt-DAC

admin ROLE_ADMIN

LIST OF THEFT USERS

You have 2 new notification

- HArinath tried a security breach
- harinath tried a security breach

See all notifications

Show 10 entries

Theft Permissions	Victim Name	Victim ID	Victim Designation	Victim Permissions	Theft Type	Theft Description	Theft Date	Status	Actions
MODIFY	hithesh	CDAC2	SOFTWARE ENGINEER	ACCESS	SEVERE	Data Thief has more permissions than the Victim, No use.	27/02/2021	Y	Image
MODIFY	hitheshjalaparthi	CDAC2	SOFTWARE ENGINEER	ACCESS	SEVERE	Data Thief has more permissions than the Victim, No use.	27/02/2021	Y	Image

localhost:8080/cryptdac/redirectpage?page=admin/viewTheftInformation#

Type here to search

10:52 28-02-2021

Photos - theft (3).jpg

See all photos + Add to

Edit & Create Share



Type here to search

11:32 28-02-2021

REFERENCES

- [1] J. Bethencourt, A. Sahai, and B. Waters, Ciphertext-policy attribute based encryption, in IEEE S&P, 2007.
- [2] X. Wang, Y. Qi, and Z. Wang, Design and Implementation of SecPod: A Framework for Virtualization-based Security Systems, IEEE Transactions on Dependable and Secure Computing, vol. 16, no. 1, 2019.
- [3] J. Ren, Y. Qi, Y. Dai, X. Wang, and Y. Shi, AppSec: A Safe Execution Environment for Security Sensitive Applications, in ACM VEE, 2015.
- [4] V. Goyal, A. Jain, O. Pandey, and A. Sahai, Bounded ciphertext policy attribute based encryption, in ICALP, 2008.
- [5] V. Goyal, O. Pandey, A. Sahai, and B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in ACM CCS, 2006.
- [6] J. Katz, A. Sahai, and B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, in EUROCRYPT, 2008.
- [7] S. Muller and S. Katzenbeisser, Hiding the policy in cryptographic access control, in STM, 2011.
- [8] R. Ostrovsky, A. Sahai, and B. Waters, Attribute-based encryption with non-monotonic access structures, in ACM CCS, 2007.
- [9] A. Sahai, and B. Waters, Fuzzy identity-based encryption, in EUROCRYPT, 2005.
- [10] T. Ring, Cloud computing hit by celebgate, <http://www.scmagazineuk.com/cloud-computing-hit-by-celebgate/article/370815/>, 2015.

- [11] X. Jin, R. Krishnan, and R. S. Sandhu, A unified attribute-based access control model covering DAC, MAC and RBAC, in DDBSec, 2012.
- [12] W. C. Garrison III, A. Shull, S. Myers, and, A. J. Lee, On the Practicality of Cryptographically Enforcing Dynamic Access Control Policies in the Cloud, in IEEE S&P, 2016.
- [13] R. S. Sandhu, Rationale for the RBAC96 family of access control models, in ACM Workshop on RBAC, 1995.
- [14] T. Jiang, X. Chen, Q. Wu, J. Ma, W. Susilo, and W. Lou, Secure and Efficient Cloud Data Deduplication With Randomized Tag, IEEE Transactions on Information Forensics and Security, vol. 12, no. 3, 2017.
- [15] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, K. Fu, Plutus: Scalable Secure File Sharing on Untrusted Storage, in USENIX FAST, 2003.
- [16] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, Verifiable Auditing for Outsourced Database in Cloud Computing, IEEE Transactions on Computers, vol. 64, no. 11, 2015.
- [17] J. Wang, X. Chen, J. Li, J. Zhao, and J. Shen, Towards achieving flexible and verifiable search for outsourced database in cloud computing, Future Generation Computer Systems, vol. 67, 2017.
- [18] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, New Publicly Verifiable Databases with Efficient Updates, IEEE Transactions on Dependable and Secure Computing, vol. 12, no. 5, 2015.
- [19] T. Jiang, X. Chen, and J. Ma, Public Integrity Auditing for Shared Dynamic Cloud Data with Group User Revocation, IEEE Transactions on Computers, vol. 65, no. 8, 2016.

[20] D. Boneh and M. Franklin, Identity-based encryption from the Weil pairing, SIAM Journal on Computing, vol. 32, no. 3, 2003.