CHAPTER 1
INTRODUCTION

1.1. SCOPE OF THE PROJECT

Cloud storage is a model of networked storage system where data is stored in pools of storage which are generally hosted by third parties. There are many benefits to use cloud storage. The most notable is data accessibility. Data stored in the cloud can be accessed at any time from any place as long as there is network access. Storage maintenance tasks, such as purchasing additional storage capacity, can be offloaded to the responsibility of a service provider. Another advantage of cloud storage is data sharing between users. If Alice wants to share a piece of data (e.g., a video) to Bob, it may be difficult for her to send it by email due to the size of data. Instead, Alice uploads the file to a cloud storage system so that Bob can download it at any time. Despite its advantages, outsourcing data storage also increases the attack surface area at the same time. For example, when data is distributed, the more locations it is stored the higher risk it contains for unauthorized physical access to the data. By sharing storage and networks with many other users it is also possible for other unauthorized users to access your data. This may be due to mistaken actions, faulty equipment, or sometimes because of criminal intent. A promising solution to offset the risk is to deploy encryption technology. Encryption can protect data as it is being transmitted to and from the cloud service. It can further protect data that is stored at the service provider. Even there is an unauthorized adversary who has gained access to the cloud, as the data has been encrypted, the adversary cannot get any information about the plaintext. Asymmetric encryption allows the encryptor to use only the public information (e.g., public key or identity of the receiver) to generate a ciphertext while the receiver uses his/her own secret key to decrypt. This is the most convenient mode of encryption for data transition, due to the elimination of key management existed in symmetric encryption.

In a normal asymmetric encryption, there is a single secret key corresponding to a public key or an identity. The decryption of ciphertext only requires this key. The key is usually stored inside either a personal computer or a trusted server or may be protected by a password. The security protection is sufficient if the computer/server is isolated from an opening network. Unfortunately, this is not what happens in the real life. When being connected with the world

through the Internet, the computer/server may suffer from a potential risk that hackers may intrude into it to compromise the secret key without letting the key owner know. In the physical security aspect, the computer storing a user decryption key may be used by another user when the original computer user (i.e., the key owner) is away (e.g., when the user goes to toilet for a while without locking the machine). In an enterprise or college, the sharing usage of computers is also common. For example, in a college, a public computer in a copier room will be shared with all students staying at the same floor. In these cases, the secret key can be compromised by some attackers who can access the victim's personal data stored in the cloud system. Therefore, there exists a need to enhance the security protection.

1.2. EXISTING SYSTEM

This is the most convenient mode of encryption for data transition, due to the elimination of key management existed in symmetric encryption. If the user has lost his security device, then his/her corresponding ciphertext in the cloud cannot be decrypted forever! That is, the approach cannot support security device update/ revocability.

As cloud computing becomes more mature and there will be more applications and storage services provided by the cloud, it is easy to foresee that the security for data protection in the cloud should be further enhanced. They will become more sensitive and important, as if the e-banking analogy. Actually, we have noticed that the concept of two-factor encryption, which is one of the encryption trends for data protection1, has been spread into some real-world applications, for example, full disk encryption with Ubuntu system, AT&T two factor encryption for Smartphones2, electronic vaulting and druva - cloud-based data encryption3. However, these applications suffer from a potential risk about factor revocability that may limit their practicability.

Disadvantages of Existing System:

1. If the user has lost his security device, then his/ her corresponding ciphertext in the cloud cannot be decrypted forever! That is, the approach cannot support security device update/revocability.
2. The sender needs to know the serial number/ public key of the security device, in additional to the user's identity/public key. That makes the encryption process more complicated.

1.3. PROPOSED SYSTEM

Our system is an IBE (Identity-based encryption)- based mechanism. That is, the sender only needs to know the identity of the receiver in order to send an encrypted data (ciphertext) to him/her. No other information of the receiver (e.g. public key, certificate etc.) is required. Then the sender sends the ciphertext to the cloud where the receiver can download it at any time.

Our system provides two-factor data encryption protection. In order to decrypt the data stored in the cloud, the user needs to possess two things. First, the user needs to have his/her secret key which is stored in the computer. Second, the user needs to have a unique personal security device which will be used to connect to the computer (e.g., USB, Bluetooth, and NFC). It is impossible to decrypt the ciphertext without either piece.

More importantly, our system, for the first time, provides security device (one of the factors) revocability. Once the security device is stolen or reported as lost, this device is revoked. That is, using this device can no longer decrypt any ciphertext (corresponding to the user) in any circumstance. The cloud will immediately execute some algorithms to change the existing ciphertext to be un-decryptable by this device. While the user needs to use his new / replacement device (together with his secret key) to decrypt his/her ciphertext. This process is completely transparent to the sender.

The cloud server cannot decrypt any ciphertext at any time. We provide an estimation of the running time of our prototype to show its practicality, using some benchmark results. We also note that although there exist some naive approaches that seem to achieve our goal, that there are many limitations by each of them and thus we believe our mechanism is the first to achieve all the above-mentioned features in the literature.

Advantages of Proposed System:

1. Our solution not only enhances the confidentiality of the data, but also offers the revocability of the device so that once the device is revoked; the corresponding ciphertext will be updated automatically by the cloud server without any notice of the data owner.
2. The cloud server cannot decrypt any ciphertext at any time

CHAPTER 2

SYSTEM ANALYSIS

2.1. PRELIMINARY INVESTIGATION

The first and foremost strategy for development of a project starts from the thought of designing a mail enabled platform for a small firm in which it is easy and convenient of sending and receiving messages, there is a search engine, address book and also including some entertaining games. When it is approved by the organization and our project guide the first activity, i.e. preliminary investigation begins. The activity has three parts:

- **Request Clarification**
- **Feasibility Study**
- **Request Approval**

2.2. REQUEST CLARIFICATION

After the approval of the request to the organization and project guide, with an investigation being considered, the project request must be examined to determine precisely what the system requires.

Here our project is basically meant for users within the company whose systems can be interconnected by the Local Area Network (LAN). In today's busy schedule man need everything be provided in a readymade manner. So, taking consideration of the vastly use of the net in day-to-day life, the corresponding development of the portal came into existence.

2.3. FEASIBILITY STUDY

An important outcome of preliminary investigation is the determination that the system request is feasible. This is possible only if it is feasible within limited resource and time. The different feasibilities that have to be analyzed are

- Operational Feasibility

- Economic Feasibility
- Technical Feasibility

OPERATIONAL FEASIBILITY

Operational Feasibility deals with the study of prospects of the system to be developed. This system operationally eliminates all the tensions of the Admin and helps him in effectively tracking the project progress. This kind of automation will surely reduce the time and energy, which previously consumed in manual work. Based on the study, the system is proved to be operationally feasible.

ECONOMIC FEASIBILITY

Economic Feasibility or Cost-benefit is an assessment of the economic justification for a computer-based project. As hardware was installed from the beginning & for lots of purposes thus the cost on project of hardware is low. Since the system is a network based, any number of employees connected to the LAN within that organization can use this tool from at any time. The Virtual Private Network is to be developed using the existing resources of the organization. So, the project is economically feasible.

TECHNICAL FEASIBILITY

According to Roger S. Pressman, Technical Feasibility is the assessment of the technical resources of the organization. The organization needs IBM compatible machines with a graphical web browser connected to the Internet and Intranet. The system is developed for platform Independent environment. Java Server Pages, JavaScript, HTML, SQL server and WebLogic Server are used to develop the system. The technical feasibility has been carried out. The system is technically feasible for development and can be developed with the existing facility.

2.4. REQUEST APPROVAL

Not all request projects are desirable or feasible. Some organization receives so many project requests from client users that only few of them are pursued. However, those projects that are both feasible and desirable should be put into schedule. After a project request is approved, it cost, priority, completion time and personnel requirements are estimated and used

to determine where to add it to any project list. Frankly speaking, the approval of those above factors, development works can be launched.

2.5 FUNCTIONAL REQUIREMENTS

Functional requirements specify which outputs file should predicted from the given file they describe the relationship between input and output of the system for each functional requirement a detailed of all data inputs and their source and the range of valid inputs must be specified.

Modules

In this implementation we have 5 Modules,

1. Private Key Generator
2. Security Device Issuer
3. Sender Module
4. Receiver Module
5. Cloud Server Module

Module Description:

Private Key Generator:

It is a trusted party responsible for issuing private key of every user.

Security Device Issuer (SDI):

It is a trusted party responsible for issuing security device of every user.

Sender:

She is the sender (and the creator) of the ciphertext. She only knows the identity (e.g., email address) of the receiver but nothing else related to the receiver. After she has created the ciphertext, she sends to the cloud server to let the receiver for download.

Receiver:

He is the receiver of the ciphertext and has a unique identity (e.g., email address). The ciphertext is stored on cloud storage while he can download it for decryption. He has a private

key (stored in his computer) and a security device (that contains some secret information related to his identity). They are given by the PKG. The decryption of ciphertext requires both the private key and the security device.

Cloud server:

The cloud server is responsible for storing all ciphertext (for receiver to download). Once a user has reported lost of his security device (and has obtained a new one from the PKG), the cloud acts as a proxy to re-encrypt all his past and future ciphertext corresponding to the new device. That is, the old device is revoked.

2.6. NON-FUNCTIONAL REQUIREMENTS

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Nonfunctional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user to user commands) or accuracy (i.e. How precise are the systems numerical answers).

The key non-functional requirements are:

- Security: The system should allow a secured communication between Sender, Router and Receiver.
- Energy Efficiency: The Time consumed by the Router to transfer the File's from the Receiver.
- Reliability: The system should be reliable and must not degrade the performance of the existing system and should not lead to the hanging of the system.

2.7. HARDWARE REQUIREMENTS

- Processor - Core 2 Duo
- Speed - 2.60 GHz
- RAM - 1GB
- Hard Disk - 500 GB
- Mouse - Optical Mouse
- Monitor - LCD

2.8. SOFTWARE REQUIREMENTS

- Operating System : Windows95/98/2000/XP
- Application Server : Tomcat5.0/6.X
- Front End : HTML, Java, JSP
- Scripts : JavaScript.
- Server-side Script : Java Server Pages.
- Database : My SQL
- Database Connectivity : JDBC.

CHAPTER 3
SYSTEM DESIGN AND DEVELOPMENT

3.1. INPUT DESIGN

Input Design plays a vital role in the life cycle of software development, it requires incredibly careful attention of developers. The input design is to feed data to the application as accurate as possible. So, inputs are supposed to be designed effectively so that the errors occurring while feeding are minimized. According to Software Engineering Concepts, the input forms or screens are designed to provide to have a validation control over the input limit, range and other related validations.

This system has input screens in almost all the modules. Error messages are developed to alert the user whenever he commits some mistakes and guides him in the right way so that invalid entries are not made. Let us see deeply about this under module design.

Input design is the process of converting the user created input into a computer-based format. The goal of the input design is to make the data entry logical and free from errors. The error is in the input are controlled by the input design. The application has been developed in user-friendly manner. The forms have been designed in such a way during the processing the cursor is placed in the position where must be entered. The user is also provided with in an option to select an appropriate input from various alternatives related to the field in certain cases.

Validations are required for each data entered. Whenever a user enters an erroneous data, error message is displayed, and the user can move on to the subsequent pages after completing all the entries in the current page.

3.2. OUTPUT DESIGN

The Output from the computer is required to mainly create an efficient method of communication within the company primarily among the project leader and his team members, in other words, the administrator and the clients. The output of VPN is the system which allows the project leader to manage his clients in terms of creating new clients and assigning new projects to them, maintaining a record of the project validity and providing folder level access

to each client on the user side depending on the projects allotted to him. After completion of a project, a new project may be assigned to the client. User authentication procedures are maintained at the initial stages itself. A new user may be created by the administrator himself or a user can himself register as a new user but the task of assigning projects and validating a new user rests with the administrator only.

The application starts running when it is executed for the first time. The server has to be started and then the internet explorer in used as the browser. The project will run on the local area network so the server machine will serve as the administrator while the other connected systems can act as the clients. The developed system is highly user friendly and can be easily understood by anyone using it even for the first time.

3.3 DATA FLOW DIAGRAM

The Data Flow Diagram is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

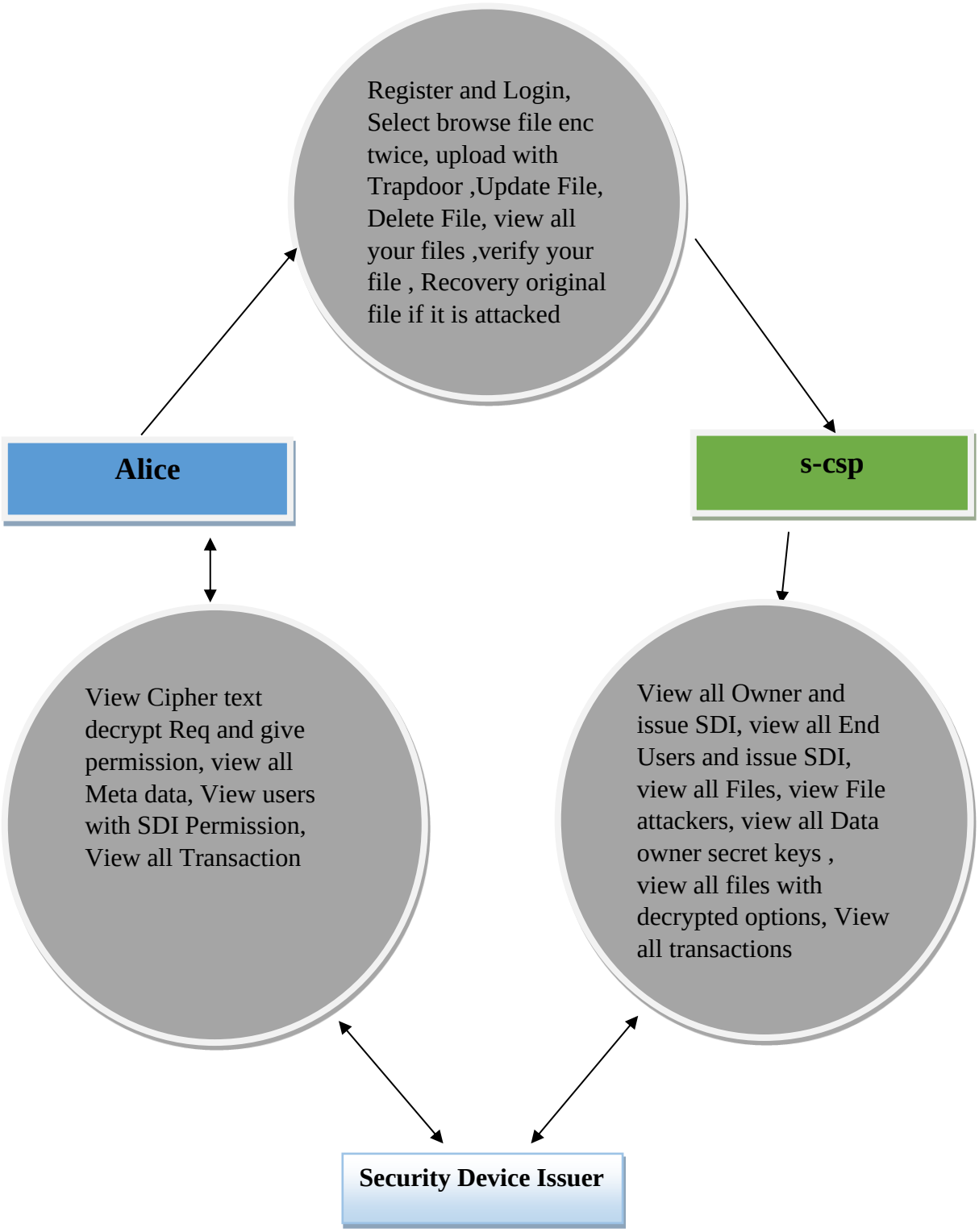
The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

DATA FLOW DIAGRAMS

Level -0



Level 1

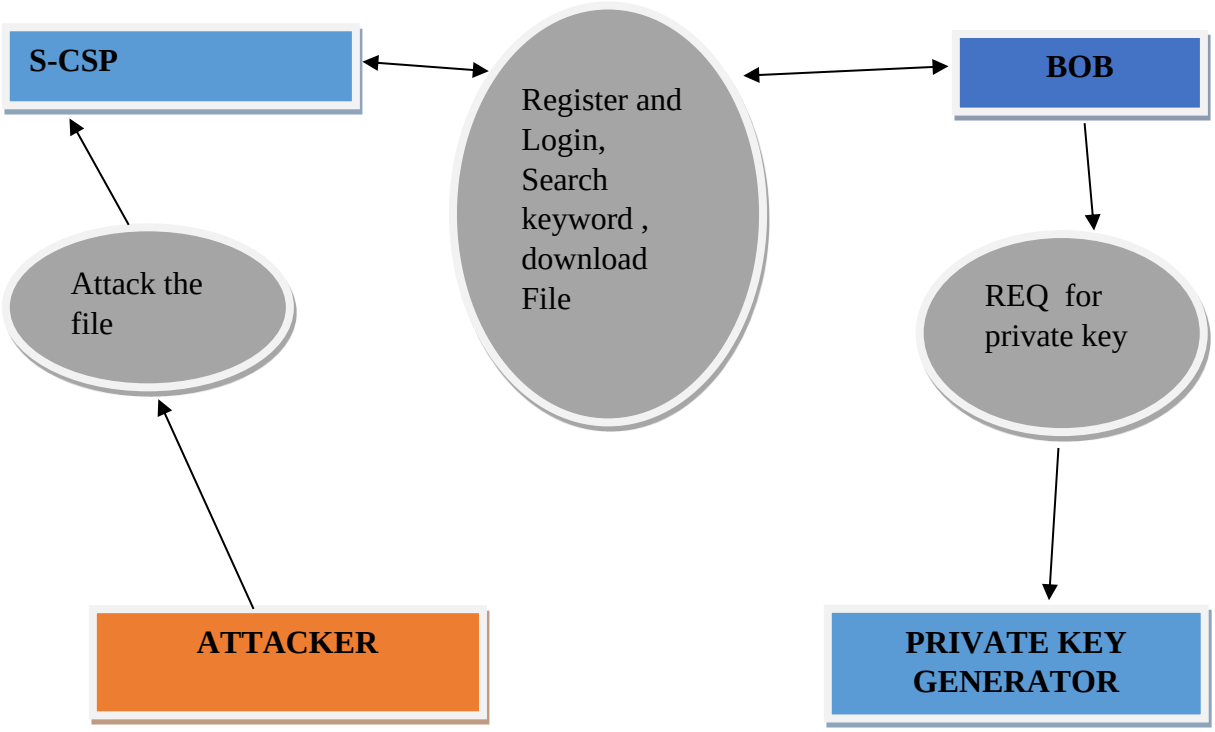


Fig 3.1 DATA FLOW DIAGRAM 1

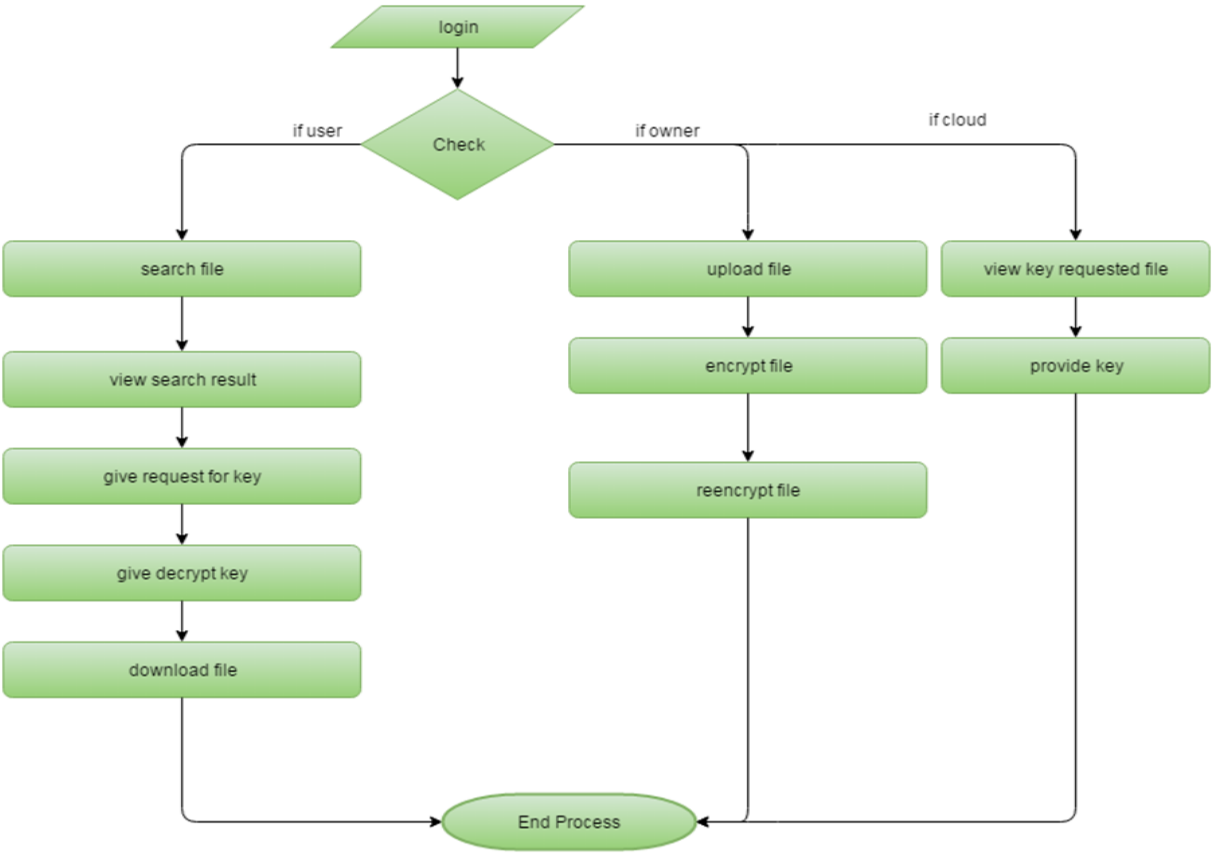


Fig 3.2 DATA FLOW DIAGRAM 2

3.4 ARCHITECTURE DIAGRAM

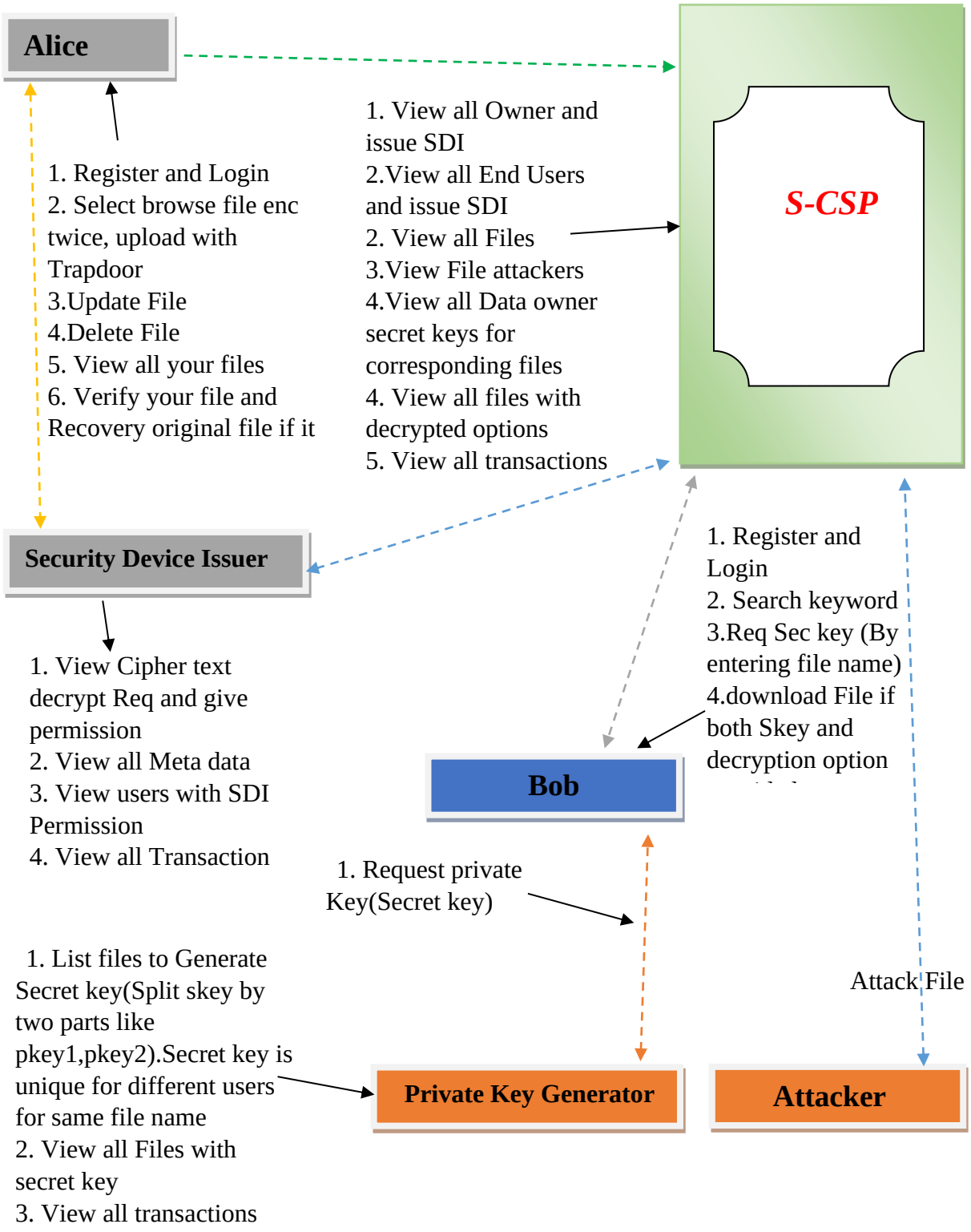


Fig 3.3 ARCHITECTURE DIAGRAM

3.5 UML DIAGRAMS

- UML is a method for describing the system architecture in detail using the blueprint.
- UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.
- UML is a very important part of developing objects oriented software and the software development process.
- UML uses mostly graphical notations to express the design of software projects.
- Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Definition:

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

UML is a language:

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

UML Specifying:

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

UML Visualization:

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

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

UML Documenting:

UML provides variety of documents in addition raw executable codes.

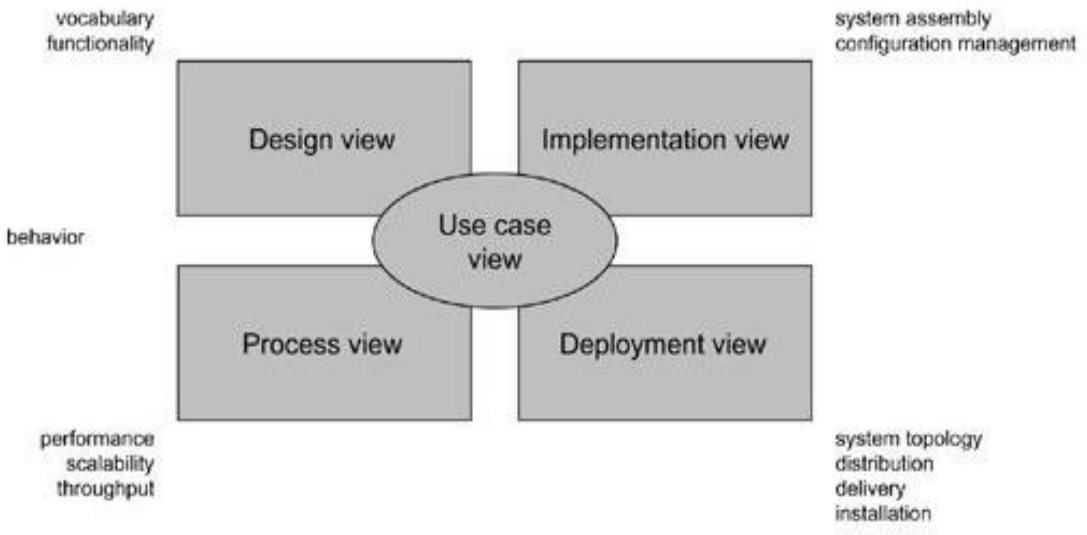


Fig: 3.4 Modeling a System Architecture using views of UML

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

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

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

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

3.5.1 USE CASE DIAGRAM

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

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

Contents:

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

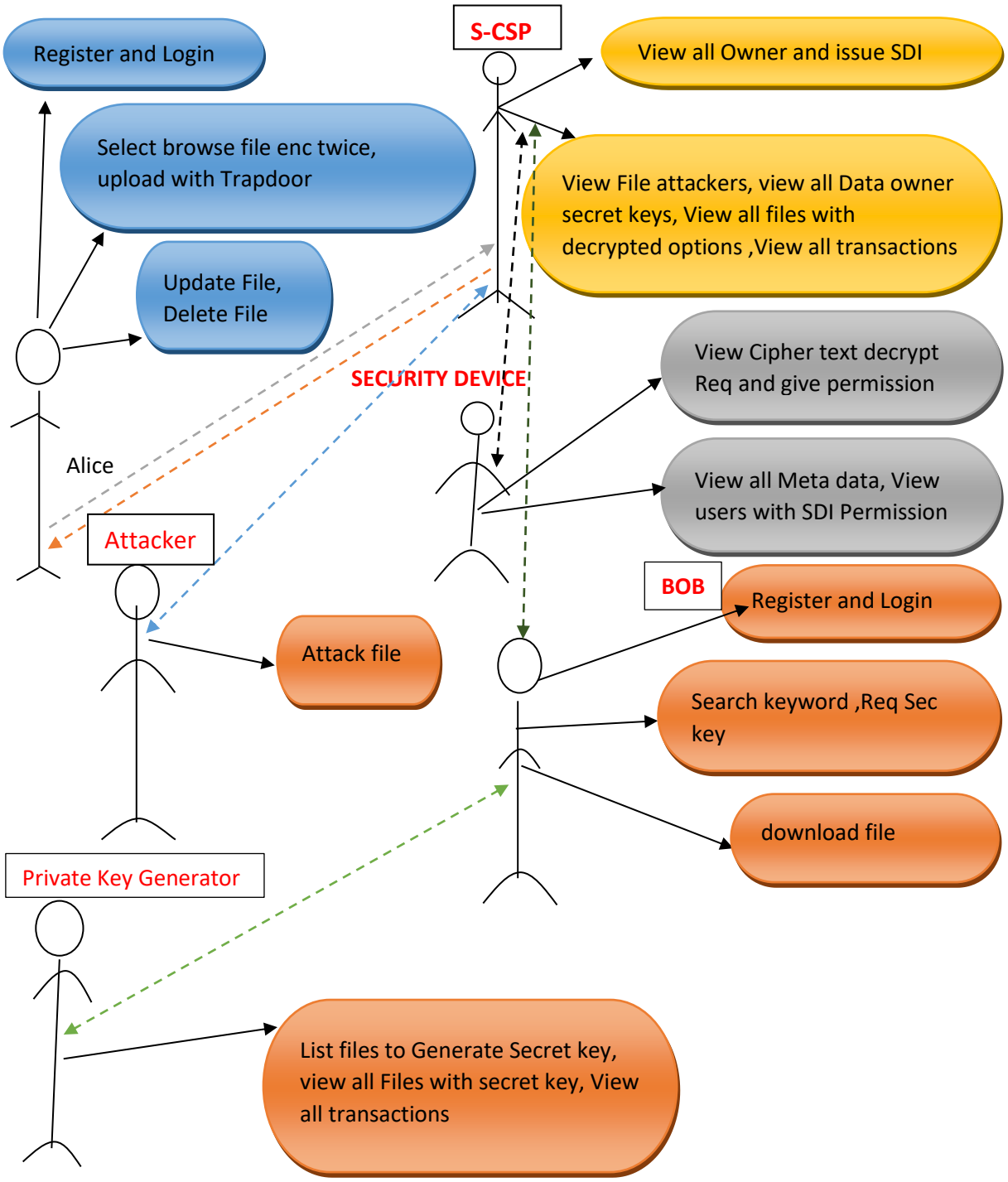


Fig 3.5 Use Case Diagram

3.5.2 CLASS DIAGRAM

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

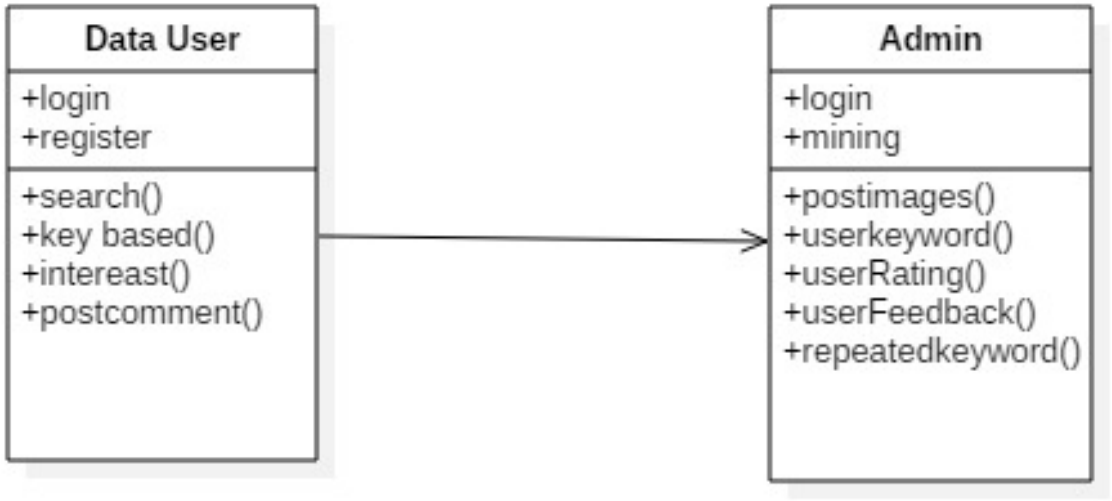


Fig: 3.6 Class Diagram 1

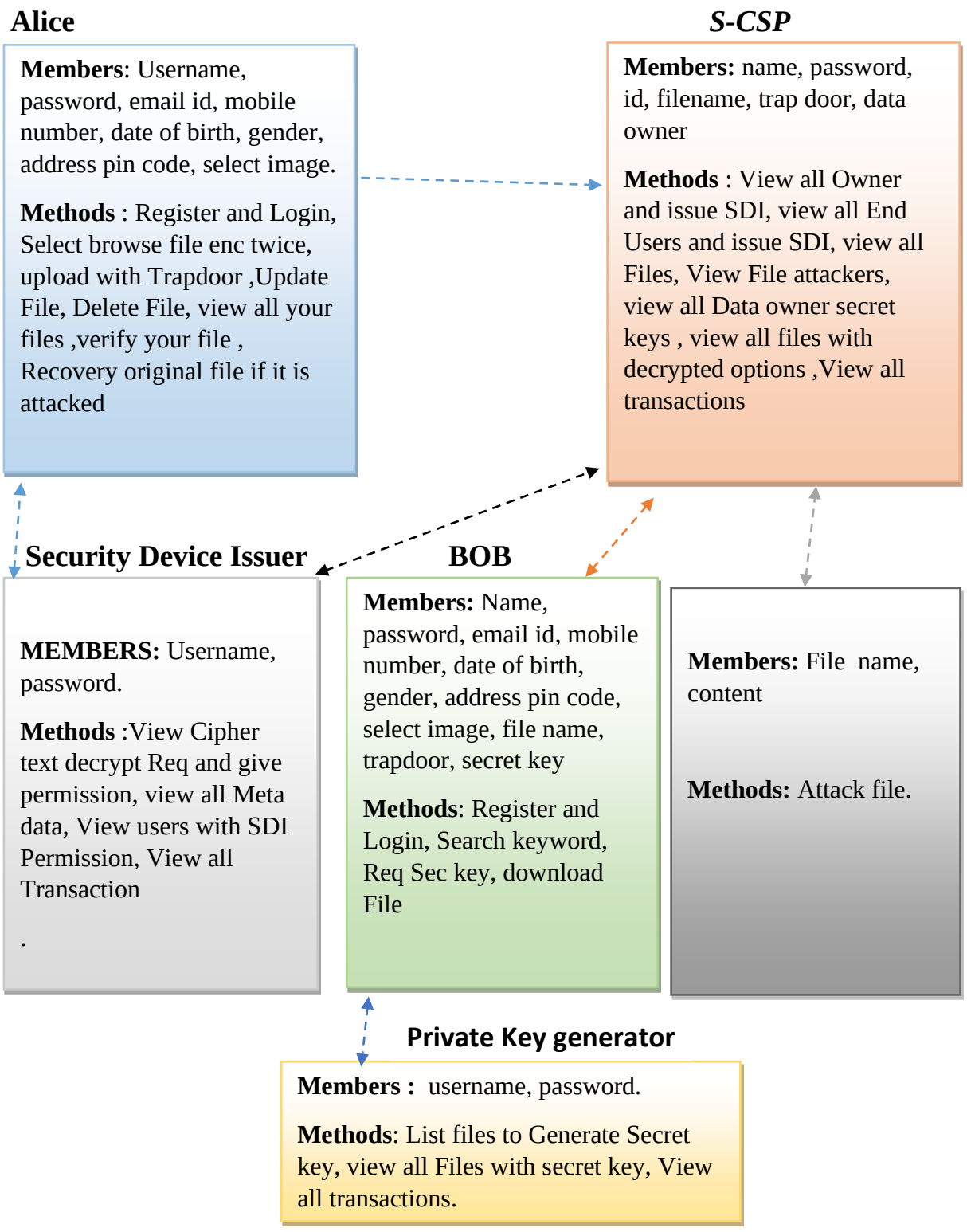


Fig 3.7 Class Diagram 2

3.5.3 SEQUENCE DIAGRAM

The processes are represented vertically and interactions are show as arrows. This article explains the purpose and the basics of Sequence diagrams.

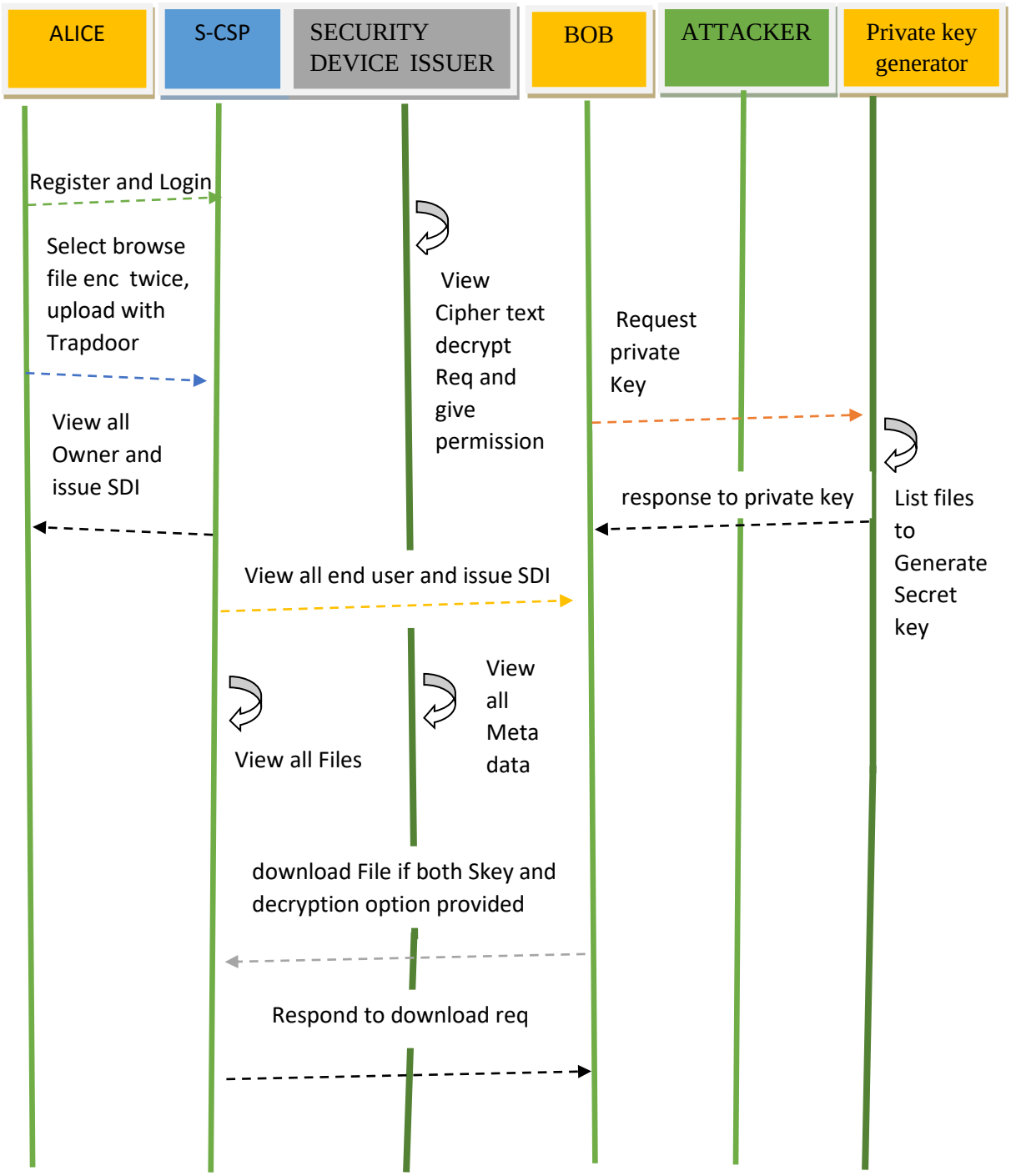


Fig 3.8 Sequence Diagram

3.5.4 ACTIVITY DIAGRAM

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

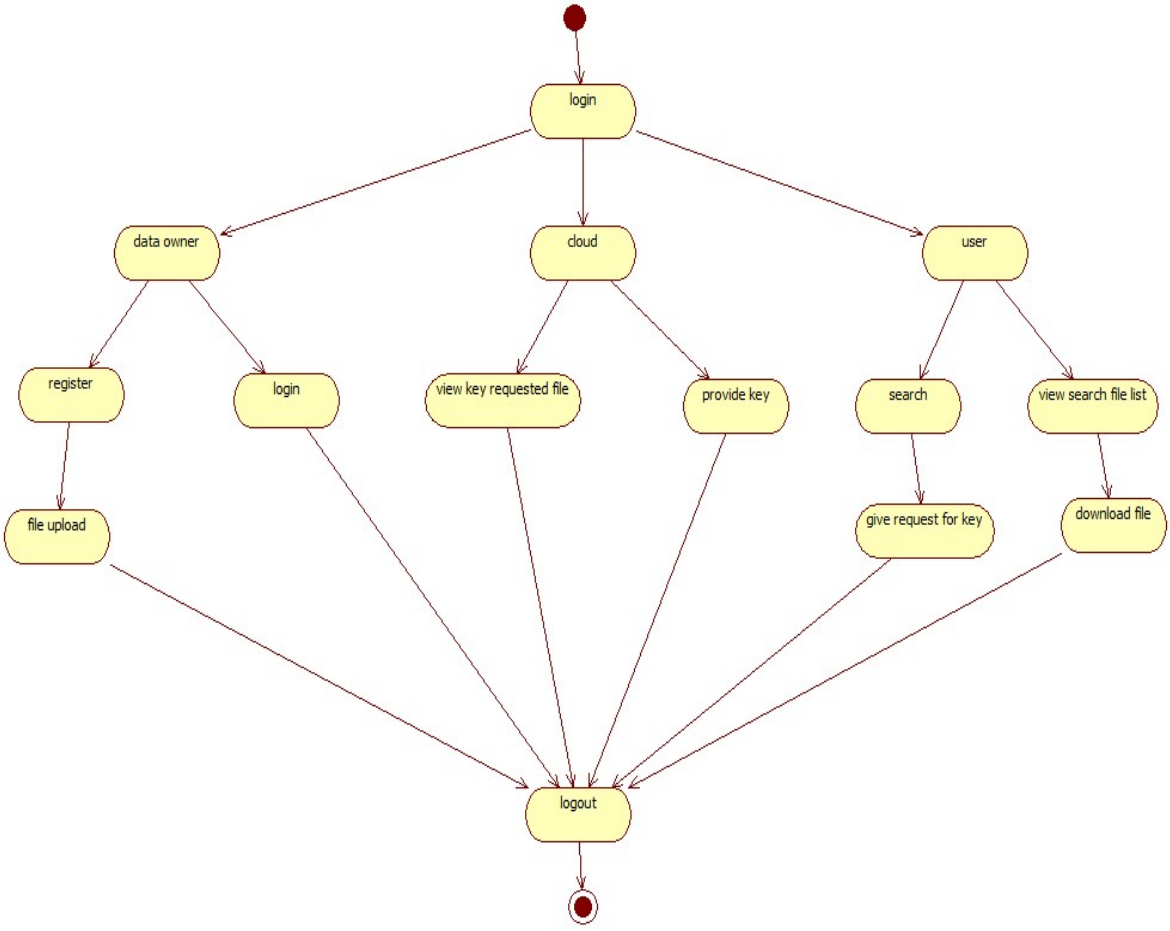


Fig:3.9 Activity Diagram for Admin

3.5.5 FLOW CHART DIAGRAM

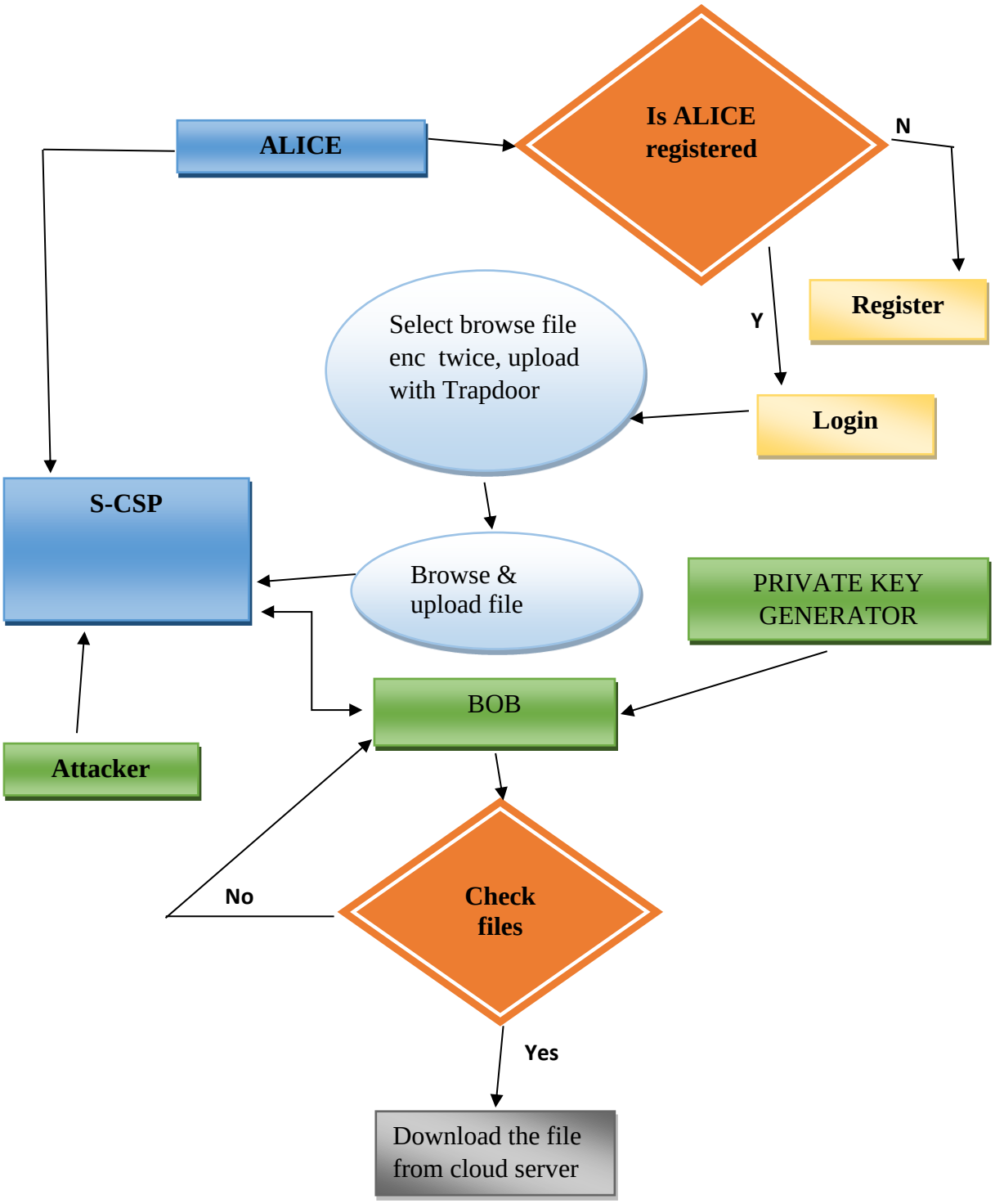


Fig 3.10 FLOW CHART

3.5.6 COMPONENT DIAGRAM

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

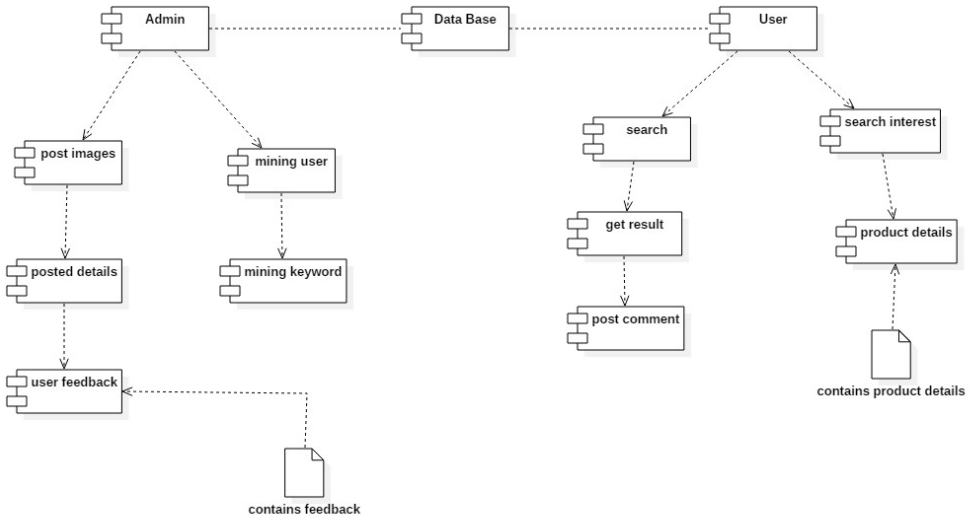


Fig:3.11. Component Diagram for Cloud

3.5.7 DEPLOYEMENT DIAGRAM

5

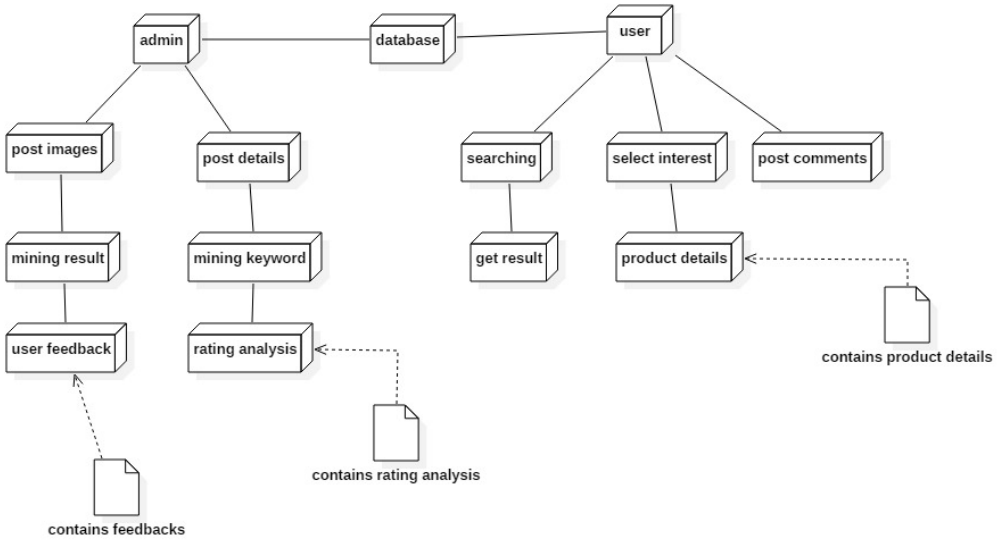


Fig:3.12. Deployment Diagram

3.6 DATABASE TABLES

3.6.1 OWNER REGISTRATION

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
Owner Name	Varchar(23)	NO	P_k	NULL	
Password	Varchar(23)	NO	P_k	NULL	
DOB	Varchar(23)	NO	P_k	NULL	
Mobile Number	Varchar(23)	NO	P_k	NULL	
Address	Varchar(11)	NO	P_k	NULL	
Email ID	Varchar(11)	NO	P_k	NULL	

Table 3.1 Owner Registration Table

3.6.2 OWNER LOGIN TABLE

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
Owner name	Varchar(23)	NO	P_k	NULL	
PWD	Varchar(23)	NO	P_k	NULL	

Table 3.2 Owner Login Table

3.6.3 USER REGISTRATION TABLE

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
Uname	Varchar(23)	NO	P_k	NULL	
Email id	Varchar(23)	NO	P_k	NULL	
Phone no	Varchar(11)	NO	P_k	NULL	
Pwd	Varchar(23)	NO	P_k	NULL	
Address	Varchar(23)	NO	P_k	NULL	

Table 3.3 User Registration Table

3.6.4 USER LOGIN TABLE

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
Username	Varchar(23)	NO	P_k	NULL	
Password	Varchar(23)	NO	P_k	NULL	

Table 3.4 User Login Table

3.6.5 FILE UPLOAD TABLE

FIELD	TYPE	NULL	KEY	DEFAULT	EXTRA
File Id	Varchar(23)	NO	P_k	NULL	
File Name	Varchar(100)	NO	P_k	NULL	
Index Name	Varchar(23)	NO	P_k	NULL	
Owner Id	Varchar(23)	NO	P_k	NULL	
Owner Name	Varchar(23)	NO	P_k	NULL	
Encrypted Key	Varchar(23)	NO	P_k	NULL	

Table 3.5 File Upload Table

CHAPTER 4

SYSTEM ARCHITECTURE

4.1 FRAMEWORK

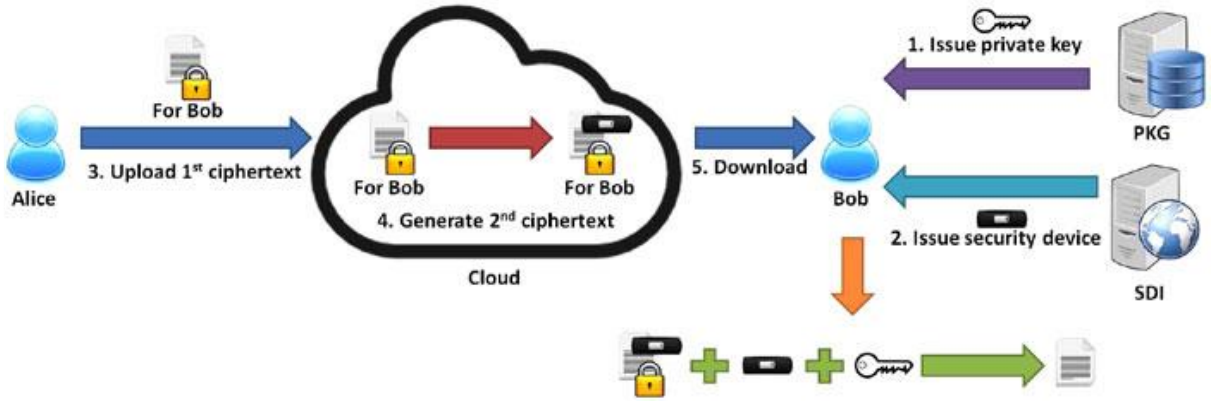


Fig 4.1 Ordinary Data Sharing

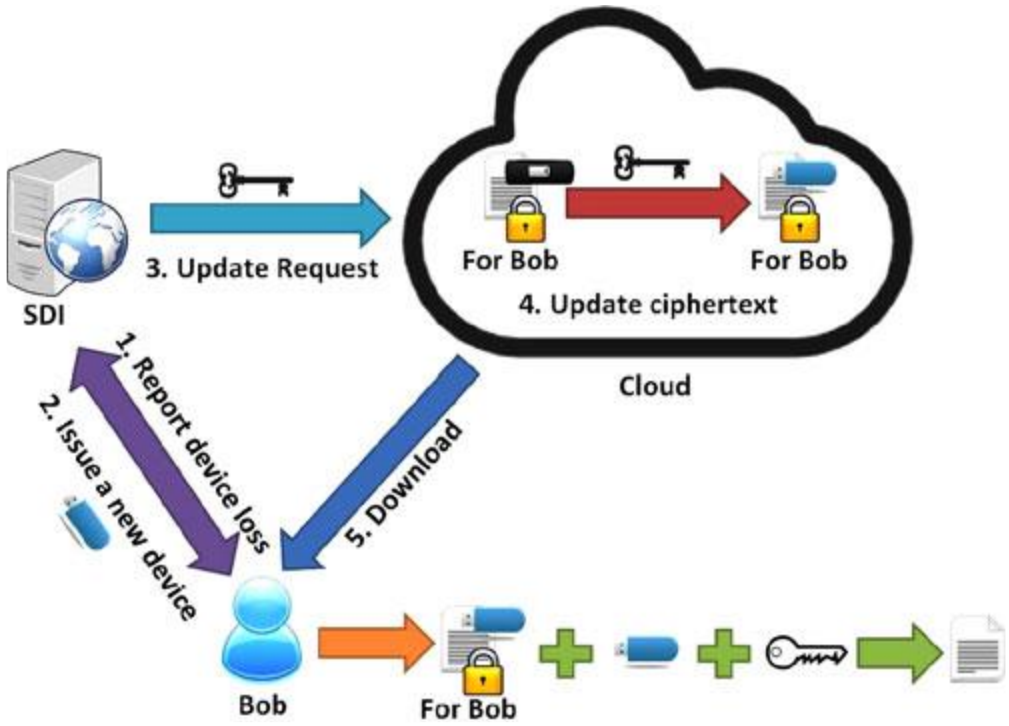


Fig 4.2 Update ciphertext after issuing a new security device

4.2 FOREGROUND AND BACKGROUND

ALICE (OWNER)

In this module sender will have to register and get authorized before he performs any operations. After the authorization, the sender can upload file with trapdoor and will have the update, delete, verify and recovery options for the file uploaded.

S-CSP

In this module S-CSP will issue SDI for both owner (Alice) and user (bob). And view the file uploaded and the attackers related to files in cloud. View the files in decrypted format and with the corresponding secret keys and its transactions.

BOB (USER)

In this module, User has to register and login, and search for the files by entering keyword and request secret key and download the particular file from the cloud if both secret key and the decryption permissions are provided.

SECURITY DEVICE ISSUER

Views all the files decrypt permission request form the users and provide permission and view its related metadata and the transactions related to the requests from users.

PRIVATE KEY GENERATOR

In this module the private key generator generates the secret key. It splits the key into two parts such as pkey1 and pkey2. This generated key is unique for different users for same file and view all the generated secret keys and the transactions related to it.

MODULE DESCRIPTION:

1. Cryptosystems with Two Secret Keys
2. Cryptosystems with Online Authority
3. Cryptosystem with Security Device
4. Cryptosystem with Revocability

Cryptosystems with Two Secret Keys

There are two kinds of cryptosystems that requires two secret keys for decryption. They are certificateless cryptosystem and certificate-based cryptosystem. Certificateless cryptosystem (CLC) was first introduced in further improvements can be found. It combines the merits of identity-based cryptosystem (IBC) and the traditional public-key infrastructure (PKI). In a CLC, a user with an identity chooses his own user secret key and user public key. At the same time the authority (called the Key Generation Centre (KGC)) further generates a partial secret key according to his identity. Encryption or signature verification requires the knowledge of both the public key and the user identity. On the opposite, decryption or signature generation requires the knowledge of both the user secret key and the partial secret key given by the KGC. Different from the traditional PKI, there is no certificate required. Thus the costly certificate validation process can be eliminated. However, the encryptor or the signature verifier still needs to know the user public key. It is less convenient than IBC where only identity is required for encryption or signature verification.

Cryptosystems with Online Authority

Mediated cryptography was first introduced for the purpose of revocation of public keys. It requires an online mediator, referred to a SEM (Security Mediator), for every transaction. The SEM also provides a control of security capabilities. If the SEM does not cooperate then no transactions with the public key are possible any longer. In other words, any revoked user cannot get the cooperation from the SEM. That means revoked users cannot decrypt any ciphertext successfully. Later on, this notion was further generalized as security mediated certificateless (SMC) cryptography. In a SMC system, a user has a secret key, public key and an identity. The user secret key and the SEM are required to decrypt a ciphertext or sign a message. On the opposite side, the user public key and the corresponding identity are needed for signature verification or encryption. Since the SEM is controlled by the revocation authority, the authority can refuse to provide any cooperation for revoked user so that no revoked user can generate signature or decrypt ciphertext. Note that SMC is different from our concept. The main purpose of SMC is to solve the revocation problem. Thus, the SME is controlled by the authority and it has to be online for every signature signing and ciphertext decryption. Furthermore, it is not identity-based. The encryptor (or signature verifier) needs to know the corresponding public key in addition to the identity. That makes the system less

practical and loses the advantages of using identity-based system.

Cryptosystem with Security Device

There is a physically secure but computationally limited device in the system. A long-term key is stored in this device, while a short-term secret key is kept by users on a powerful but insecure device where cryptographic computations take place. Short term secrets are then refreshed at discrete time periods via interaction between the user and the base while the public key remains unchanged throughout the lifetime of the system. The user obtains a partial secret key from the device at the beginning of each time period. He then combines this partial secret key with the one from the previous period, in order to renew the secret key for the current time period.

Different from our concept, key-insulated cryptosystem requires all users to update their key in every time period. It may require some costly time synchronization algorithms between users which may not be practical in many scenarios. The key update process requires the security device. Once the key has been updated, the signing or decryption algorithm does not require the device anymore within the same time period. While our concept does require the security device every time the user tries to decrypt the ciphertext.

Cryptosystem with Revocability

Another cryptosystem supporting revocability is proxy re-encryption (PRE). Decryption rights delegation is introduced in Blaze, Bleumer and Strauss formally defined the notion of PRE. To employ PRE in the IBE setting, Green and Ateniese defined the notion of identity-based PRE (IB-PRE). Later on, Tang, Hartel and Jonker proposed a CPA-secure IB-PRE scheme, in which delegator and delegatee can belong to different domains. After that there are many IB-PRE systems have been proposed to support different user requirements. Among of the previously introduced IB-PRE systems, is the most efficient one without loss of revocability. We state that leveraging can only achieve one of our design goals, revocability, but not two-factor protection.

CHAPTER 5

IMPLEMENTATION

5.1 INTRODUCTION TO TECHNOLOGY USED

Java History :

Java is a programming language created by James Gosling from Sun Microsystems in 1991. The first publicly available version of Java (Java 1.0) was released in 1995. Over time new enhanced versions of Java have been released. The current version of Java is Java 1.7 which is also known as Java 7.

From the Java programming language, the Java platform evolved. The Java platform allows that the program code is written in other languages than the Java programming language and still runs on the Java virtual machine.

Java Virtual Machine :

The Java virtual machine (JVM) is a software implementation of a computer that executes programs like a real machine. The Java virtual machine is written specifically for a specific operating system, e.g. for Linux a special implementation is required as well as for Windows.

Java programs are compiled by the Java compiler into so-called byte code. The Java virtual machine interprets this byte code and executes the Java program.

Java Runtime Environment Vs. Java Development Kit :

Java comes in two flavors, the Java Runtime Environment (JRE) and the Java Development Kit (JDK). The Java runtime environment (JRE) consists of the JVM and the Java class libraries and contains the necessary functionality to start Java programs. The JDK contains in addition the development tools necessary to create Java programs. The JDK consists therefore of a Java compiler, the Java virtual machine, and the Java class libraries.

The Java Development Kit (JDK) is provided by Sun Microsystems as a basic development environment for Java. The JDK provides similar facilities to the cc compiler for

C programs, plus a JVM simulator and some additional facilities such as a debugger. To use the JDK, programs are constructed as acai text files (by using an editor, for example). The program files are compiled, which translates the Java code to JVM byte code in .class files.

Each public class must be in a file having the class name (case sensitive on Unix) followed by a .java suffix. There may be any number of classes defined in a .java file, but the compiler produces a separate .class file for each class. A file is compiled with the java command, which is similar to the cc (or gecko) command. A class is executed (or more precisely, the method main in a class is executed) by the command java with the class name (not the .class file) as the parameter. Thus, for example, to compile the program in file Hi.java, we would use the command

java Hi.java

And then to execute the program we would use the command

Java Hi

Both compile-time and execution-time (exceptions) error messages include the file name and line where the error occurred. No .class file is produced if there is a compile-time error.

Characteristics of Java :

The target of Java is to write a program once and then run this program on multiple operating systems.

Java has the following properties:

- Platform independent: Java programs use the Java virtual machine as abstraction and do not access the operating system directly. This makes Java programs highly portable. A Java program which is standard complaint and follows certain rules can run unmodified on all supported platforms, e.g., Windows or Linux.
- Object-orientated programming language: Except the primitive data types, all elements in Java are objects.
- Strongly typed programming language: Java is strongly typed, e.g. the types of the used variables must be pre-defined and conversion to other objects is relatively strict, e.g. must be done in most cases by the programmer.

- Interpreted and compiled language: Java source code is transferred into the byte code format which does not depend on the target platform. These byte code instructions will be interpreted by the Java Virtual machine (JVM). The JVM contains a so-called Hotspot-Compiler which translates performance critical byte code instructions into native code instructions.
- Automatic memory management: Java manages the memory allocation and de-allocation for creating new objects. The program does not have direct access to the memory. The so-called garbage collector deletes automatically objects to which no active pointer exists.

The Java syntax is similar to C++. Java is case sensitive, e.g. the variables my Value and my value will be treated as different variables.

Development Process with Java :

The programmer writes Java source code in a text editor which supports plain text. Normally the programmer uses an Integrated Development Environment (IDE) for programming. An IDE supports the programmer in the task of writing code, e.g. it provides auto-formatting of the source code, highlighting of the important keywords, etc.

At some point the programmer (or the IDE) calls the Java compiler (javac). The Java compiler creates the byte code instructions. . These instructions are stored in .class files and can be executed by the Java Virtual Machine.

J2SE :

J2SE is a collection of Java Programming Language API (Application programming interface) that is very useful to many Java platform programs. It is derived from one of the most dynamic programming language known as “JAVA”

J2SE is a collection of Java Programming Language API (Application programming interface) that is very useful to many Java platform programs. It is derived from one of the most dynamic programming language known as “JAVA” and one of its three basic editions of Java known as Java standard edition being used for writing Applets and other web based applications.

J2SE platform has been developed under the Java umbrella and primarily used for writing applets and other Java-based applications. It is mostly used for individual computers. Applet is a type of fast-working subroutine of Java that is platform-independent but work within other frameworks. It is a mini application that perform a variety of functions, large and small, ordinary and dynamic, within the framework of larger applications.

J2SE provide the facility to users to see Flash movies or hear audio files by clicking on a Web page link. As the user clicks, page goes into the browser environment and begin the process of launching application-within-an-application to play the requested video or sound application. So many online games are being developed on J2SE. JavaBeans can also be developed by using J2SE.

J2EE：

Java***2*** Platform **E**nterprise **E**dition. J2EE is a platform -independent, Java -centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multitier, Web-based applications.

Key features and services of J2EE :

- At the client tier, J2EE supports pure HTML, as well as Java applets or applications. It relies on Java Server Pages and servlet code to create HTML or other formatted data for the client.
- Enterprise JavaBeans (EJBs) provide another layer where the platform's logic is stored. An EJB server provides functions such as threading, concurrency, security and memory management. These services are transparent to the author.
- Java Database Connectivity (JDBC), which is the Java equivalent to ODBC, is the standard interface for Java databases.
- The Java servlet API enhances consistency for developers without requiring a graphical user interface.

JDBC :

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks commonly associated with database usage:

- Making a connection to a database
- Creating SQL or MySQL statements
- Executing that SQL or MySQL queries in the database
- Viewing & Modifying the resulting records

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as:

- Java Applications
- Java Applets
- Java Servlets
- Java ServerPages (JSPs)
- Enterprise JavaBeans (EJBs)

All of these different executables are able to use a JDBC driver to access a database and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

JDBC Architecture :

The JDBC API supports both two-tier and three-tier processing models for database access but in general JDBC Architecture consists of two layers:

- **JDBC API:** This provides the application-to-JDBC Manager connection.
- **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases. The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application.

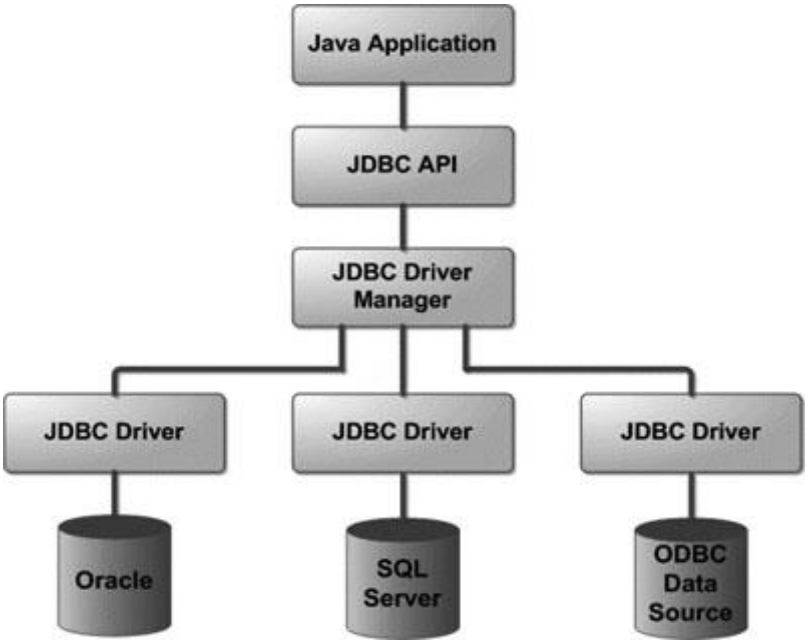


Fig 5.1 JDBC Architecture

Common JDBC Components :

The JDBC API Provides the Following Interfaces and Classes :

- **Driver Manager:** This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication subprotocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
- **Driver:** This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager

objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects

- **Connection :** This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
- **Statement :** You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
- **ResultSet:** These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
- **SQLException:** This class handles any errors that occur in a database application.

JDBC Driver

JDBC drivers implement the defined interfaces in the JDBC API for interacting with your database server. For example, using JDBC drivers enable you to open database connections and to interact with it by sending SQL or database commands then receiving results with Java.

The Java.sql package that ships with JDK contains various classes with their behaviors defined and their actual implementations are done in third-party drivers. A third-party vendor implements the java.sql.Driver interface in their database driver.

JDBC Drivers Types

JDBC driver implementations vary because of the wide variety of operating systems and hardware platforms in which Java operates. Sun has divided the implementation types into four categories, Types 1, 2, 3, and 4, which is explained below:

Type 1: JDBC-ODBC Bridge Driver

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC requires configuring on your system a Data Source Name (DSN) that represents the target database.

When Java first came out, this was a useful driver because most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available.

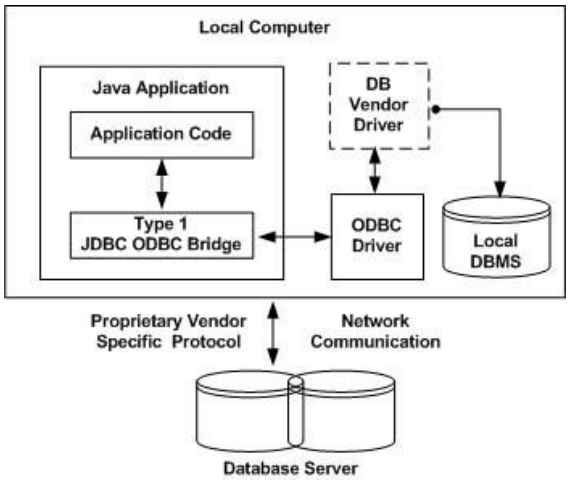


Fig 5.2 JDBC-ODBC Bridge Driver

The JDBC-ODBC bridge that comes with JDK 1.2 is a good example of this kind of driver.

Type 2: JDBC-Native API

In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls which are unique to the database. These drivers typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge, the vendor-specific driver must be installed on each client machine. If we change the Database we have to change the native API as it is specific to a database and they are mostly obsolete now but you may realize some speed increase with a Type 2 driver, because it eliminates ODBC's overhead.

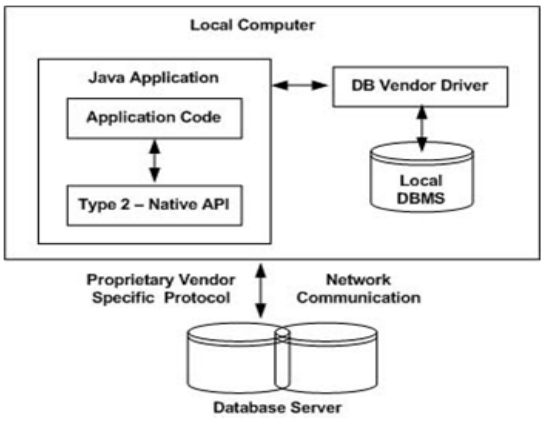


Fig. 5.3 JDBC-Native API

Type 3: JDBC-Net pure Java

In a Type 3 driver, a three-tier approach is used to accessing databases. The JDBC clients use standard network sockets to communicate with an middleware application server. The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server. This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.

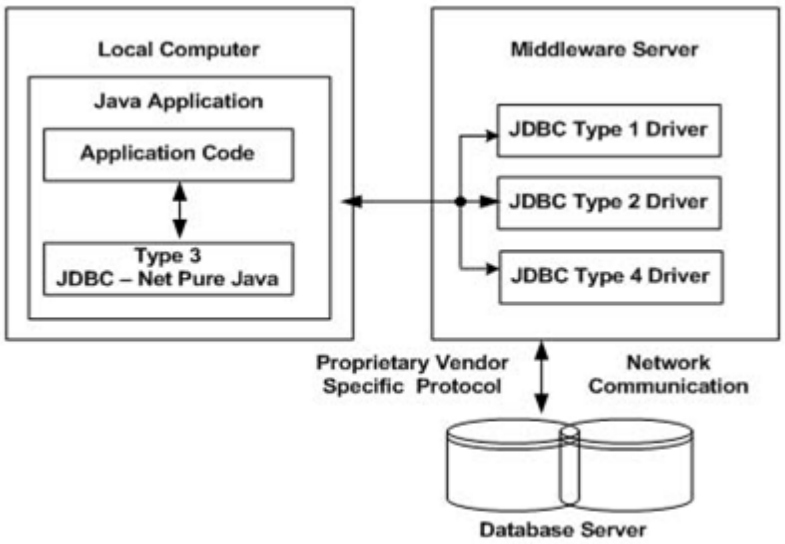


Fig 5.4 JDBC-Net pure Java 1

You can think of the application server as a JDBC "proxy," meaning that it makes calls for the client application. As a result, you need some knowledge of the application server's configuration in order to effectively use this driver type.

Your application server might use a Type 1, 2, or 4 driver to communicate with the database, understanding the nuances will prove helpful.

Type 4: 100% pure Java

In a Type4 driver, a pure Java-based driver that communicates directly with vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself. This kind of driver is extremely flexible,

you don't need to install special software on the client or server. Further, these drivers can be downloaded dynamically.

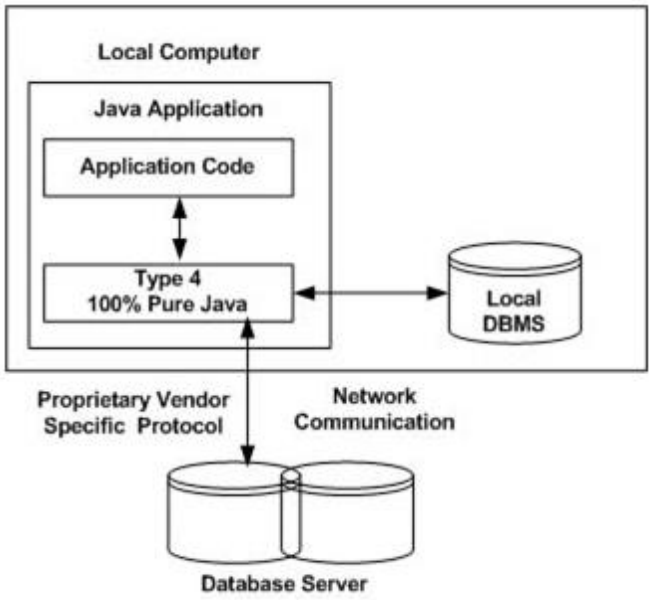


Fig 5.5 JDBC-Net pure Java 2

MySQL's Connector/J driver is a Type 4 driver. Because of the proprietary nature of their network protocols, database vendors usually supply type 4 drivers.

JDBC Database Connections :

After you've installed the appropriate driver, it's time to establish a database connection using JDBC.

The programming involved to establish a JDBC connection is fairly simple. Here are these simple four steps:

- **Import JDBC Packages:** Add import statements to your Java program to import required classes in your Java code.
- **Register JDBC Driver:** This step causes the JVM to load the desired driver implementation into memory so it can fulfill your JDBC requests.
- **Database URL Formulation:** This is to create a properly formatted address that points to the database to which you wish to connect.

- **Create Connection Object:** Finally, code a call to the DriverManager object's getConnection( ) method to establish actual database connection.

Import JDBC Packages :

The Import statements tell the Java compiler where to find the classes you reference in your code and are placed at the very beginning of your source code.

To use the standard JDBC package, which allows you to select, insert, update, and delete data in SQL tables, add the following *imports* to your source code:

```
import java.sql.* ;  // for standard JDBC programs

import java.math.* ; // for BigDecimal and BigInteger support
```

Register JDBC Driver :

You must register your driver in your program before you use it. Registering the driver is the process by which the Oracle driver's class file is loaded into memory so it can be utilized as an implementation of the JDBC interface.

5.2 SOFTWARE ENVIRONMENT

Java Technology :

Java technology is both a programming language and a platform.

The Java Programming Language:

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

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust

- Dynamic
- Secure

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

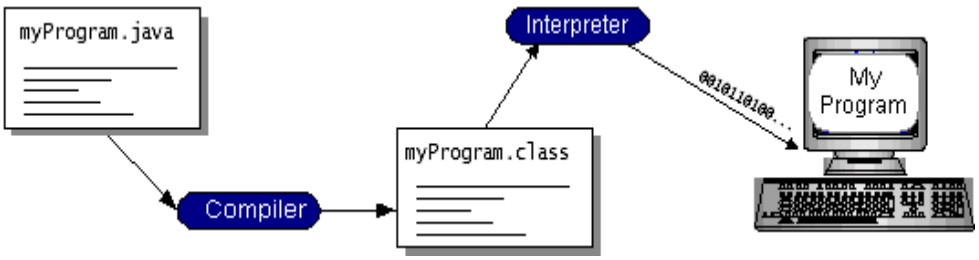


Fig 5.6 Java Compiler

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

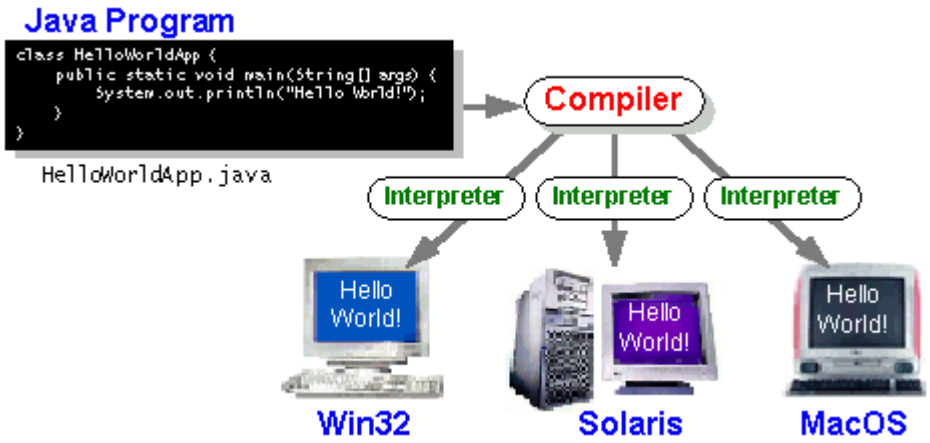


Fig 5.7 Java virtual machine

The Java Platform :

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

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

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

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

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

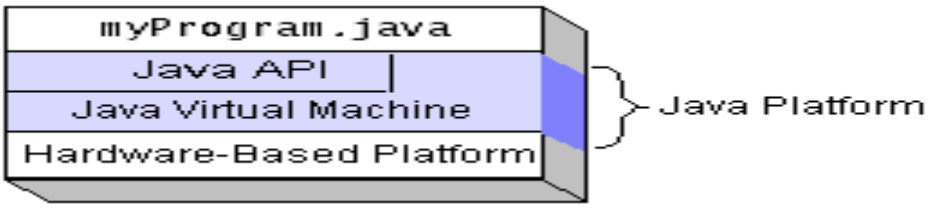


Fig 5.8 Java Application Programming Interface (Java API)

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

- **Java Database Connectivity (JDBC™)**: Provides uniform access to a wide range of relational databases. The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

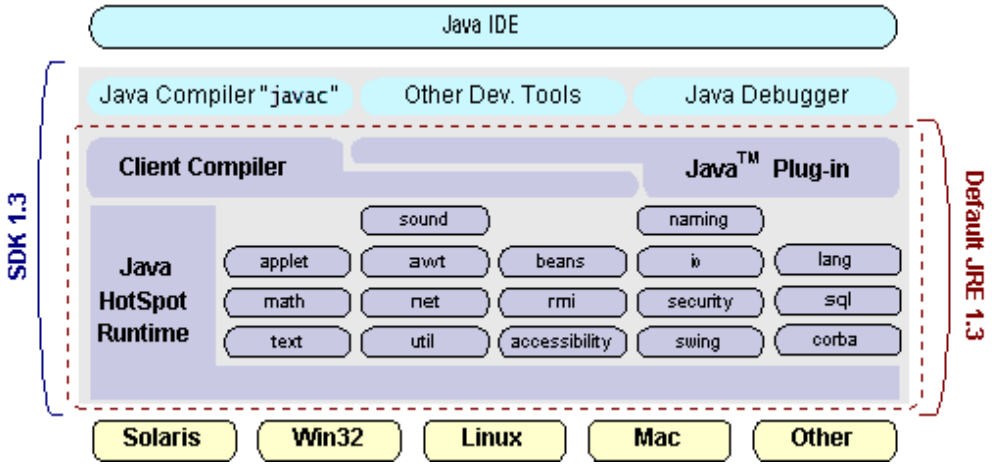


Fig 5.9 Java 2 SDK architecture

SQLLevelAPI :

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to "generate" JDBC code and to hide many of JDBC's complexities from the end user.

SQL Conformance :

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

JDBC Must Be Implemental On Top Of Common Database Interfaces. The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

Networking :

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP Datagram's :

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP :

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP :

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet Addresses :

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network Address :

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16-bit network addressing. Class C uses 24-bit network addressing and class D uses all 32.

Subnet Address :

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host Address :

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total Address : The 32-bit address is usually written as 4 integers separated by dots.

Port Addresses :

A service exists on a host and is identified by its port. This is a 16-bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets :

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call `socket`. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with `Read File` and `Write File` functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int typ e, int protocol);
```

Here "family" will be `AF_INET` for IP communications, `protocol` will be zero, and `type` will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

Tomcat 6.0 web server

Tomcat is an open-source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and Java Server Pages technologies. The Java Servlet and Java Server Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components (BEAs Weblogic, is one of the popular application server).To develop a web application with jsp/servlet install any web server like JRun, Tomcat etc. to run your application.

CHAPTER 6

TESTING

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

6.1 TESTING METHODOLOGIES

The following are the Testing Methodologies:

- **Unit Testing.**
- **System Testing**
- **White Box Testing**
- **Black Box Testing**
- **Integration Testing**
- **Output Testing**
- **Validation Testing**

Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually, and the module interfaces are verified for the consistency with design specification. All-important processing path are tested for the expected results. All error handling paths are also tested.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a

configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

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

Black Box Testing

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

Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

The following are the types of Integration Testing:

1. Top-Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required

for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom-up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom-up approaches test each module individually and then each module is module is integrated with a main module and tested for functionality.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

Validation Testing

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data

information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

Preparation of Test Data

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Using Live Test Data:

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true system test and in fact ignores the cases most likely to cause system failure.

USER TRAINING

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is extremely easy.

MAINTAINENCE

This covers a wide range of activities including correcting code and design errors. To

reduce the need for maintenance in the long run, we have more accurately defined the user's requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

6.2 TEST CASES

Test Case 1 : Admin login (successful)

Test case number	Test case	Input	Expected output	Obtained output
1	User Login	Give Username and Password	User Page Open	User Page Open

Table 6.1 Test Case for User Login

Test case number	Test case	Input	Expected output	Obtained output
1	Owner Login	Give patient name and password	Owner page is open	Owner page is open

Table 6.2 Test Case for Owner Login

CHAPTER 7

SAMPLE CODE

```
<%@ page import="java.sql.*, databaseconnection.*"%>
<%@ page import="java.io.*"%>
<%@ page import = "java.util.Date,java.text.SimpleDateFormat,java.text.ParseException"%>
<html>
<head>
<title>revisiting</title>
</head>
<body>
<%
//String email=(String)session.getAttribute("email");
//                                System.out.println("email in encrypt db page is"+ email);
String re4=(String)session.getAttribute("re2");
System.out.println("re4="+re4);
System.out.println("recrypt db in encryptdb page is"+re4);
String a=(String)session.getAttribute("fname");
System.out.println("email in encryptdb page is"+a);
String passw=(String)session.getAttribute("passw");
System.out.println("email in encrypt page is"+passw);
java.util.Date now = new java.util.Date();
String date=now.toString();
String DATE_FORMAT = "yyyy-mm-dd";
SimpleDateFormat sdf = new SimpleDateFormat(DATE_FORMAT);
String strDateNew = sdf.format(now) ;
// String pkey="waiting for key";
ResultSet rs = null;
try
{
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/twofactor",
"root", "");
PreparedStatement ps = con.prepareStatement("INSERT INTO reencrypt VALUES(?,?,?,?)");
```

```
ps.setString(1,a);
ps.setString(2,passw);
ps.setString(3,strDateNew);
ps.setString(4,re4);
ps.executeUpdate();
response.sendRedirect("index.html?Success");
}
catch(Exception x)
{
out.println(x.getMessage());
}
%>
</body>
</html>
```

CHAPTER 8

OUTPUT SCREEN

8.1 HOME PAGE

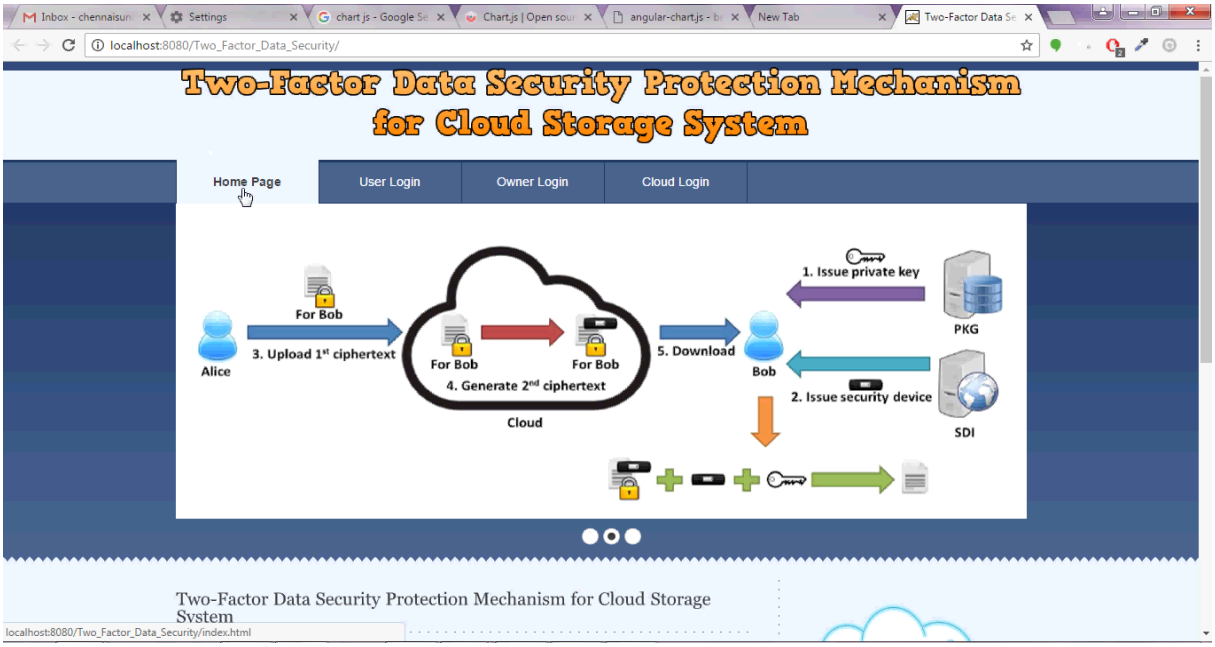


Fig:8.1 Home Page

8.2 USER LOGIN PAGE

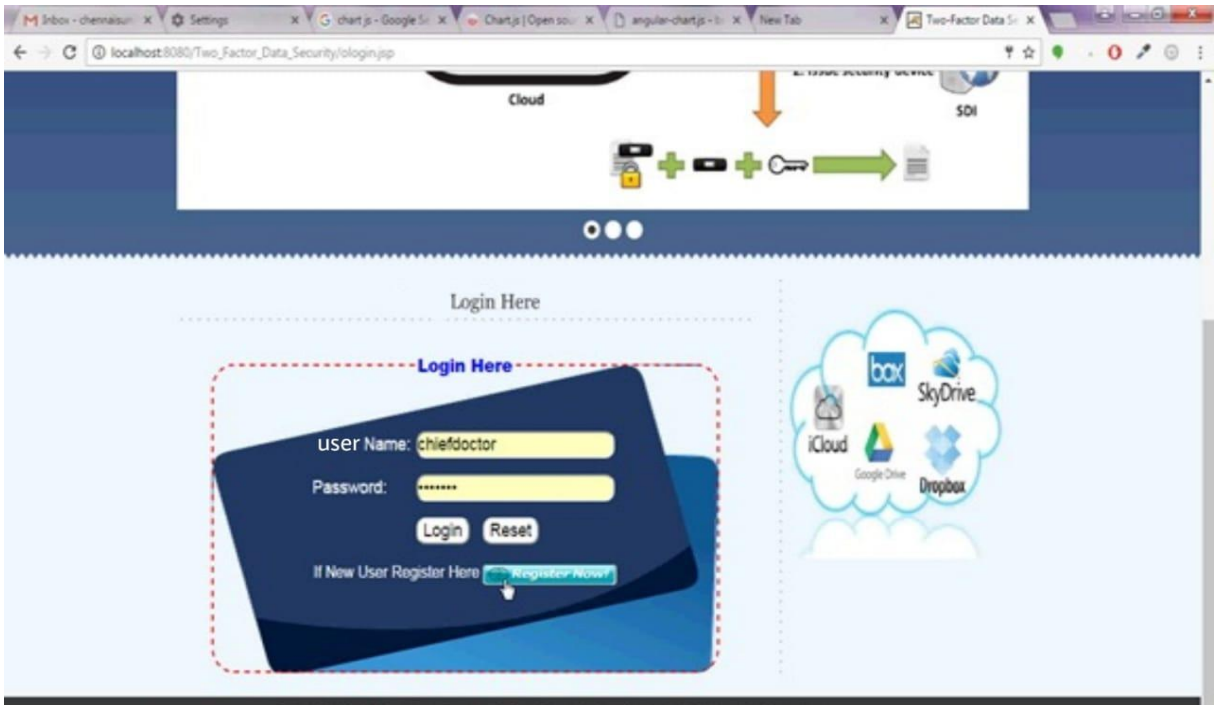


Fig:8.2 User Login Page

8.3 OWNER LOGIN PAGE

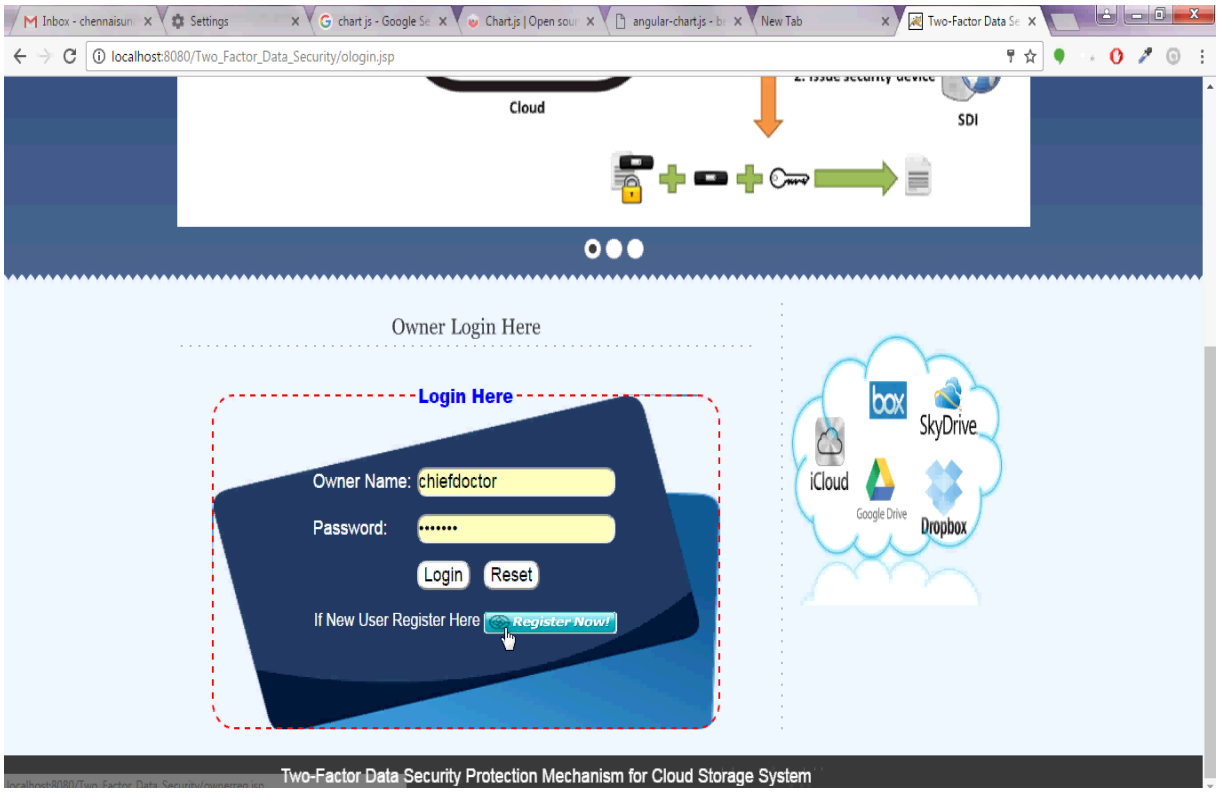


Fig:8.3 Owner Login Page

8.4 OWNER REGISTRATION PAGE

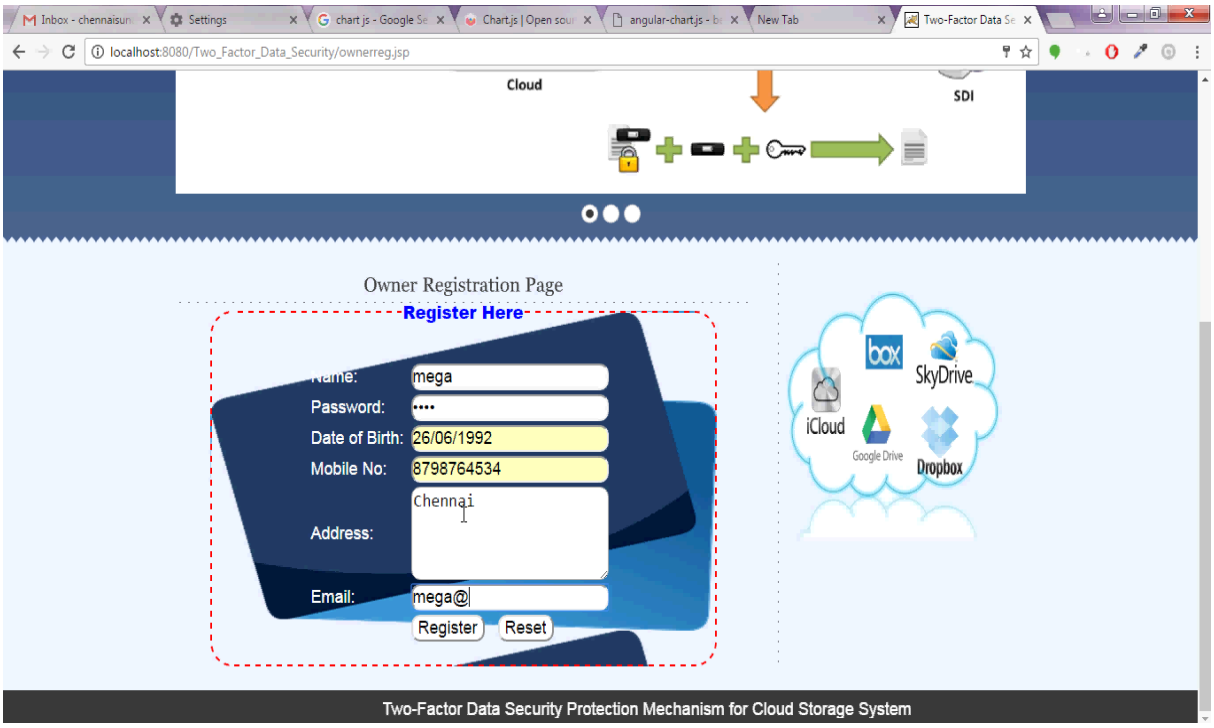


Fig:8.4 Owner Registration Page

8.5 OWNER FILE UPLOAD HERE

Fig:8.5 Owner file Upload Here

8.6 USER REGISTRATION PAGE

Fig:8.6 User Registration Page

8.7 SEARCH PAGE

Fig:8.7 Search Page

8.8 SEARCH RESULT

Fig:8.8 Search Result

8.9 CLOUD LOGIN PAGE

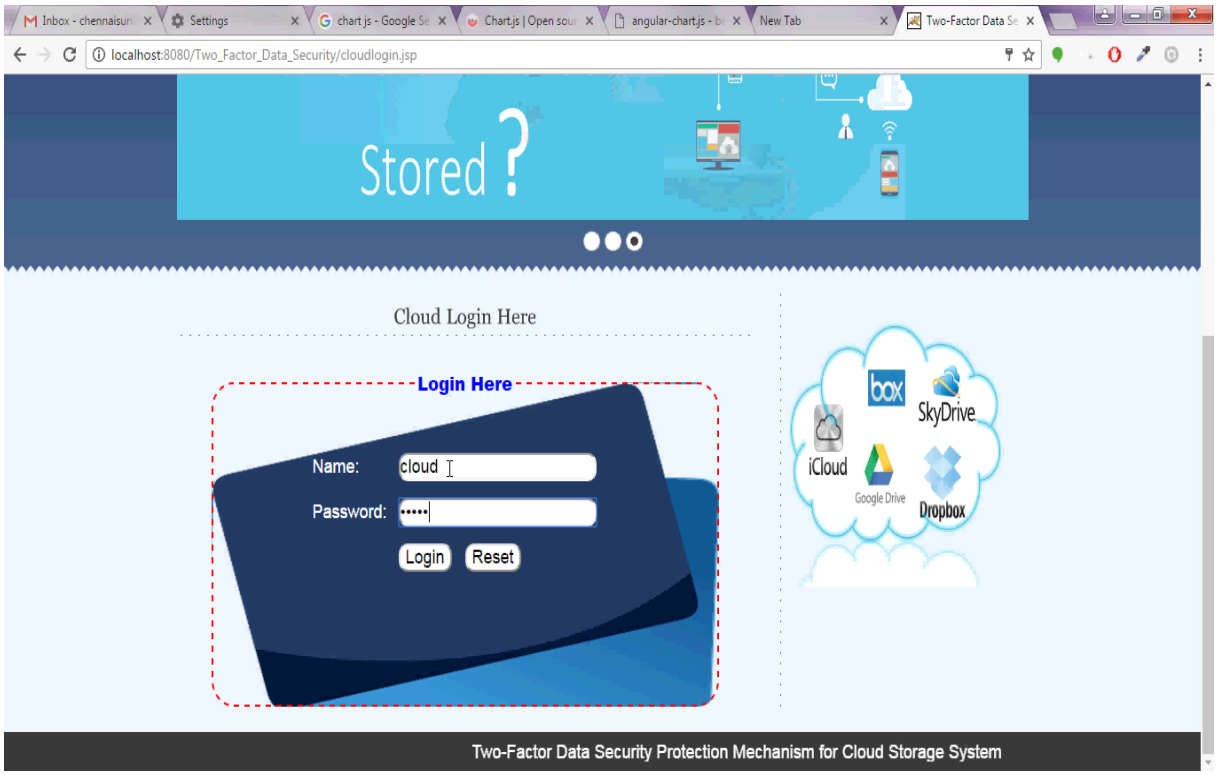


Fig:8.9 Cloud Login Page

8.10 USER KEY REQUESTED FILE LIST

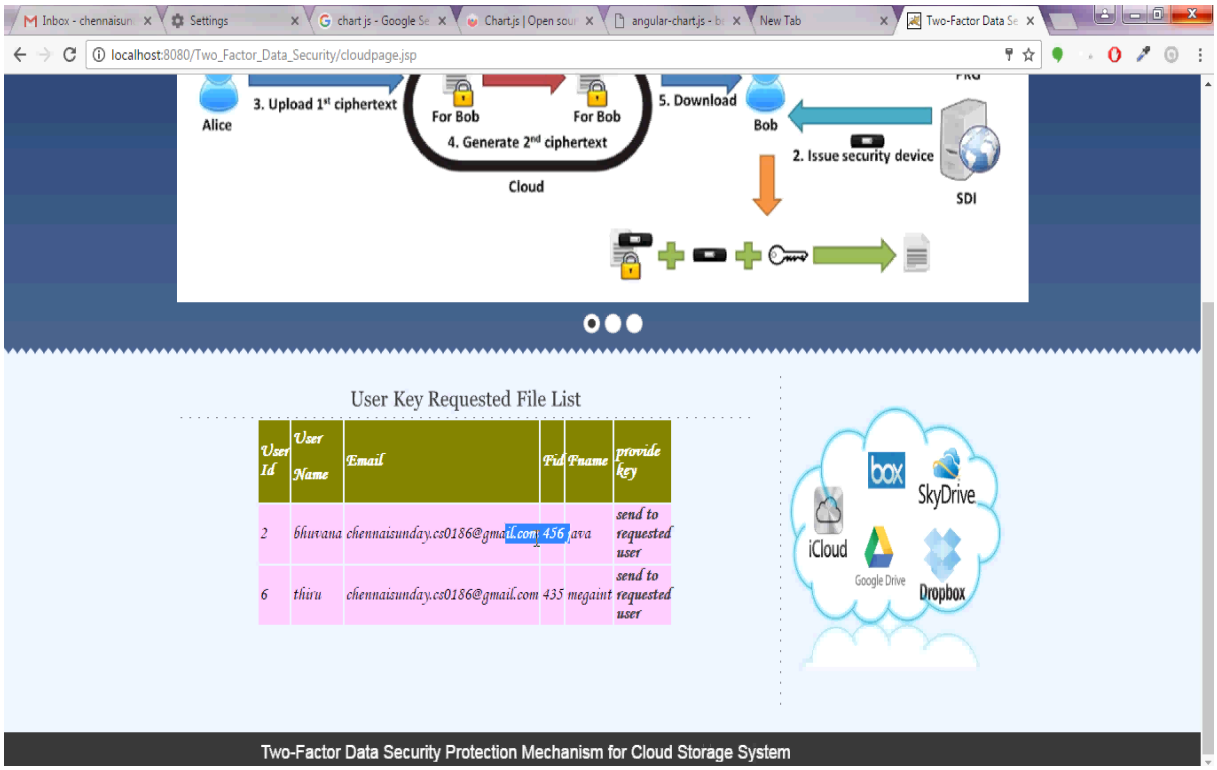


Fig:8.10 User Key Requested File List

CHAPTER 9

CONCLUSION

In this paper, we introduced a novel two-factor data security protection mechanism for cloud storage system, in which a data sender is allowed to encrypt the data with knowledge of the identity of a receiver only, while the receiver is required to use both his/her secret key and a security device to gain access to the data. Our solution not only enhances the confidentiality of the data, but also offers the revocability of the device so that once the device is revoked, the corresponding ciphertext will be updated automatically by the cloud server without any notice of the data owner. Furthermore, we presented the security proof and efficiency analysis for our system.

REFERENCES

[1] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In TCC, volume 5444 of Lecture Notes in Computer Science, pages 474–495. Springer, 2009.

[2] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In ASIACRYPT, volume 2894 of Lecture Notes in Computer Science, pages 452–473. Springer, 2003.

[3] M. H. Au, J. K. Liu, W. Susilo, and T. H. Yuen. Certificate based (linkable) ring signature. In ISPEC, volume 4464 of Lecture Notes in Computer Science, pages 79–92. Springer, 2007.

[4] M. H. Au, Y. Mu, J. Chen, D. S. Wong, J. K. Liu, and G. Yang. Malicious kgc attacks in certificateless cryptography. In ASIACCS, pages 302–311. ACM, 2007.

[5] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In K. Nyberg, editor, EUROCRYPT, volume 1403 of LNCS, pages 127–144. Springer, 1998.

[6] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based encryption with efficient revocation. In P. Ning, P. F. Syverson, and S. Jha, editors, ACM Conference on Computer and Communications Security, pages 417–426. ACM, 2008.

[7] D. Boneh, X. Ding, and G. Tsudik. Fine-grained control of security capabilities. ACM Trans. Internet Techn., 4(1):60–82, 2004.

[8] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In CRYPTO ’01, volume 2139 of LNCS, pages 213– 229. Springer, 2001.

[9] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, ACM Conference on Computer and Communications Security, pages 185–194. ACM, 2007.

[10] H. C. H. Chen, Y. Hu, P. P. C. Lee, and Y. Tang. Nccloud: A network-coding-based storage system in a cloud-of-clouds. IEEE Trans. Computers, 63(1):31–44, 2014.

[11] S. S. M. Chow, C. Boyd, and J. M. G. Nieto. Security-mediated certificateless cryptography. In Public Key Cryptography, volume 3958 of Lecture Notes in Computer Science, pages 508–524. Springer, 2006.

[12] C.-K. Chu, S. S. M. Chow, W.-G. Tzeng, J. Zhou, and R. H. Deng. Key-aggregate cryptosystem for scalable data sharing in cloud storage. IEEE Trans. Parallel Distrib. Syst., 25(2):468–477, 2014.