VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

"Efficient Privacy-Aware Authentication Scheme For Mobile Cloud"

Thesis submitted in partial fulfillment of the curriculum prescribed for the award of the degree of Bachelor of Engineering in Computer Science & Engineering by

1CR14CS067	Kumar Amber
1CR14CS079	Mayank Roshan
1CR14CS065	Kishlay Bhagwan
1CR14CS028	Ayussh Soarav

Under the Guidance of

Navaneetha. M
Assistant professor
Department of CSE, CMRIT, Bengaluru

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
#132, AECS LAYOUT, IT PARK ROAD, BENGALURU - 560037

2017-18

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI - 590018

Certificate

This is to certify that the project entitled "Efficient Privacy-Aware Authentication Scheme For Mobile Cloud " is a bonafide work carried out by Mayank Roshan, Kumar Amber, Kishlay Bhagwan, and Ayussh Soarav in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2017-18. It is certified that all corrections / suggestions indicated during reviews have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

----------	----------	----------
Signature of Guide	Signature of HoD	Signature of Principal
Navaneetha. M	**Dr. Jhansi Rani P**	**Dr. Sanjay Jain**
Assistant Professor	Professor & Head	Principal
Department of CSE	Department of CSE	CMRIT,
CMRIT, Bengaluru - 37	CMRIT, Bengaluru - 37	Bengaluru - 37

External Viva

	Name of the Examiners	Institution	Signature with Date
1.	----------	----------	----------
2.	----------	----------	----------

Acknowledgement

We take this opportunity to thank all of those who have generously helped us to give a proper shape to our work and complete our BE project successfully. A successful project is fruitful culmination efforts by many people, some directly involved and some others indirectly,by providing support and encouragement.

We would like to thank Dr. SANJAY JAIN , Principal , CMRIT ,for providing excellent academic environment in the college.

We would like to express our gratitude towards Dr. JHANSI RANI , Professor and HOD , Dept of CSE , CMRIT , who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our Internal Guide Ms. NAVANEETHA M , Assistant Professor , Department of Computer Science and Engineering , CMRIT , for her valuable guidance throughout the tenure of this project work.

Amber
Mayank
Ayussh
Kishlay

Table of Contents

List of Figures

Abstract

With the exponential increase of the mobile devices and the fast development of cloud computing, a new computing paradigm called mobile cloud computing (MCC) is put forward to solve the limitation of the mobile devices storage, communication, and computation. Through mobile devices, users can enjoy various cloud computing services during their mobility. However, it is difficult to ensure security and protect privacy due to the openness of wireless communication in the new computing paradigm. Recently, Tsai and Lo proposed a privacy-aware authentication (PAA) scheme to solve the identification problem in MCC services and proved that their scheme was able to resist many kinds of existing attacks. Unfortunately, we found that Tsai and Los scheme cannot resist the service provider impersonation attack, i.e., an adversary can impersonate the service provider to the user. Also, the adversary can extract the users real identity during executing the service provider impersonation attack. To address the above problems, in this paper, we construct a new PAA scheme for MCC services by using an identity-based signature scheme. Security analysis shows that the proposed PAA scheme is able to address the serious security problems existing in Tsai and Los scheme and can meet security requirements for MCC services. The performance evaluation shows that the proposed PAA scheme has less computation and communication costs compared with Tsai and Los PAA scheme.

Chapter 1

Preamble

1.1 Introduction

Due to the deployment of wireless communication technologies and the popularity of mobile devices (such as laptop, intelligent mobile phone, and tablet PC), we can access the Internet services during mobility. This brings much convenience to our daily life as we can enjoy many kinds of network services anywhere and anytime. With users increasing demand of high services quality, a huge amount of data should be processed in time by his/her mobile device. However, the mobile devices resources (such as storage, computation, and communication capabilities) are limited and they cannot satisfy users requirements. This weakness has become a performance bottleneck of various applications based on mobile devices. In the past several years, the cloud computing developed rapidly as one of the powerful network technologies. Through the resource visualization technology, the cloud computing is able to provide convenient and cheap services to users in a pay-as -you-go mode. For example, we can get some cloud storage services freely from many famous cloud service providers (CSPs) such as Baidu and Google. A new digital ecosystem called the mobile cloud computing (MCC) emerged recently, where the mobile computing is integrated with cloud computing platforms. With this integration, the resource constrained problems of mobile devices could be addressed successfully. With the increase of MCC services types, the distributed MCC is also employed in practical applications, where many kinds of CSPs are able to provide different types of cloud services to users.

1.2 Existing System

To achieve mutual authentication (MA) in open networks, Lamport proposed the first authentication scheme for the single server environment. However, Lamports scheme is not able to resist the replay attack and the impersonation attack. In order to improve

security, several password-based authentication schemes are proposed. Compared with Lamports scheme, those schemes have many advantages. However, each server in those schemes has to maintain a verifier table to achieve the MA. The adversary may impersonate the user or the server when he/she steals verifier tables. To remove the serious weaknesses, it is necessary to design authentication schemes without any verifier table.

1.2.1 Disadvantages of Existing System

- Existing system is not able to resist the replay attack and the impersonation attack.
- It is necessary to maintain the verifier table in each server to achieve mutual authentication which lead to impersonation attack.
- Less secure system.

1.3 Methodology used

- DES Encyption DES is the archetypal block cipheran algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another ciphertext bitstring of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.
- RNS Algoritm: A residue numeral system is defined by a set of N integer constants,

  m1, m2, m3, ... , mN , referred to as the moduli. Let M be the least common multiple of all the mi.

  Any arbitrary integer X smaller than M can be represented in the defined residue numeral system as a set of N smaller integers

  x1, x2, x3, ... , xN with

  xi = X modulo mi representing the residue class of X to that modulus.
- SMTP Protocol: SMTP is a connection-oriented, text-based protocol in which a mail sender communicates with a mail receiver by issuing command strings and

supplying necessary data over a reliable ordered data stream channel, typically a Transmission Control Protocol (TCP) connection. An SMTP session consists of commands originated by an SMTP client (the initiating agent, sender, or transmitter) and corresponding responses from the SMTP server (the listening agent, or receiver) so that the session is opened, and session parameters are exchanged. A session may include zero or more SMTP transactions.

- FTP Protocol: The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

  FTP is built on a client-server model architecture and uses separate control and data connections between the client and the server.[1] FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS) or replaced with SSH File Transfer Protocol (SFTP).

Chapter 2

System Design

2.1 Proposed System

In this system, we present the security analysis of Tsai and Los PAA scheme. Through a concrete attack, we point that Tsai and Los PAA scheme is insecure against the service provider impersonation attack. To enhance security, we construct a new PAA scheme for the MCC services by using an identity-based signature scheme. The major contributions of this paper are summarized as follows.

1. First, we review and analyze Tsai and Los PAA scheme for the MCC services. Through a concrete attack, we show that their scheme is insecure against the service provider impersonation attack.

2. Second, we propose a new PAA scheme for the MCC services based on an identity-based signature scheme. Our proposed PAA scheme overcomes the weakness existing in Tsai and Los scheme for the MCC services.

3. Finally, we provide a detailed security and performance analysis to show that the proposed PAA scheme not only meets requirements in the MCC services.

2.1.1 Advantages

- Protect against replay attack and the impersonation attack.
- No need to maintain verifier table in each server to achieve mutual authentication
- More secure system

2.1.1.1 System Architecture

A typical network model of the PAA scheme for MCC services is shown in Fig. 1. There are three participants in a PAA scheme for MCC services: the trusted smart key generator SKG, a user Ui , and a CSPj

Figure 2.1: Network Model

SKG: It is trusted third party and is responsible for generating the master key and the system parameters. It also generates the private keys of Ui and CSPj according to their identities. Ui : He/she is a user of the system. Using the private key generated by SCG and a mobile device with limited resources, he/she could pass CSPj s authentication and access some MCC services. CSPj : It is powerful CSP and could provide some specific MCC services. Using the private key and system parameters generated by SCG, CSPj is able to check the legality of Ui , and makes Ui confirm that it is a legal CSP.

Chapter 3

Requirements

3.1 Hardware

- System : Pentium IV 2.4 GHz.
- Hard Disk : 500 GB.
- Ram : 4 GB

3.2 Software

- Operating system : Windows XP / 7
- Coding Language : Java (Jdk 1.7)
- Web Technology : Servlet, JSP
- Web Server : Tomcat 7.0
- IDE : Eclipse Galileo
- Database : My-SQL 5.1
- UGI for DB : SQLyog
- JDBC Connection : Type 4

Chapter 4

System Analysis

In the proposed system, data owner will upload the file to the cloud storage by doing the proxy re-encryption to secure the confidentiality of the file , proxy re-encryption means encrypting the encrypted data, each and every data owner will have the RNS and DES key , while uploading the file to the cloud storage file will be encrypted with the RNS key of the data owner and the encrypted file will again encrypt with the DES key of the Cloud Service Provider. Then data owner has the authority to give the access permission to which group of users can able access the file. In the proposed system more security is providing to the file, where user has to send the request to Cloud Service Provider by providing his identity token, Cloud Service Provider system has to verify and approves the request, once its approve it will fetch the file from the cloud and decrypt with DES Key and send it to User. User has to decrypt the file using RNS Key which he received from Data Owner.

4.1 Sessions

- Admin Session : Domain Authority is a super user who creates the Data Owner user and maintains the cloud servers configurations. He has the writes to Add, Edit or Delete any number of Data owners. Once the Domain Authority logged in he has following functions.
  - Cloud Server (Add, Edit, Delete)
  - Data Owner (Add, Edit, Delete)
  - Add Key Generation for both the algorithm
  - Domain (View Only)
  - Sub-Domain (View Only)
  - Change Password

- Data Owner Session : Data Owner is a person who will store the files in cloud which in turn accessed by the authorized Data Consumers. Data Owners are like Liberian who will upload all the files in the system. Whenever the file is uploaded it will be encrypted by the system using Data Owners Encryption Key (Two Layer Encryption). Data Owner has to specify the Access Policy for each and every file. Access policies are set using Domain Attribute and Sub-Domain Attribute. Once the Data Owner logged in he has following functions.
  - User Details (View, Delete)
  - View and Send Secret File
  - View All Request
  - Verify Identity Token
  - Send Secret File to requested User
  - Get RNS Key and DES Key
  - Get user Domain and Sub Domain Details
  - Concatenate Keys + Domain Details + Expiry Date
  - Encrypt the above string using DES algorithm
  - Send the Secret file to Requested User Email ID
  - File Upload
  - File Selection
  - Encrypting using RNS
  - Cloud Selection
  - Move to cloud
  - Transfer the Encrypted file to selected cloud
  - Encrypting RNS output using DES
  - Uploaded File Details (View, Delete)
  - File Access Control Setting
  - File Access Control Details (View, Delete)
  - Transaction Details
  - Change Password
- Data User : Data User are the data access users, suppose data owner is a college Liberian then data consumers are like students, lectures and admin staff in a college. Data User can able to register themselves and he will receive the

Identity Token through email. Data Consumer will receive their access key (Attributed based Decryption Key) from respective data owner through email. With the help of the access key they can able to download the files for which they have access, remember access control is set by data owner. Suppose the data consumer wants to download any file, first he has to select the file from the list and the system ask for the access key, After system getting the access key it will separate the Attribute Set from the key and check for the access rights, if the user has the access he can download the encrypted file which in turn decrypted using the decryption key and download to the data consumer local system. Once the Data Consumer logged in he has following functions.

- User Registration (Data Consumer)
- Fill the user details
- Provide Domain and Sub Domain Details
- Payment Gateway [Optional]
- Generate a Identity Token
- Email Identity Token to the user
- Login
- Identity Token Verification
- Request for Secret File
- Enter User ID
- Display User Details
- Upload Identity Token
- File Details (View)
- File Download
- Select the file from the list
- Select the Secret Identity file from the local system
- Send secret Identity file and Selected file to cloud
- Decrypt the Secret identity file
- Get the Domain Values
- Check the Access Control using Domain values
  If Access Control pass download the file or deny the file access
- Enter the transaction record in the table
- File Decrypt

- Select the file to be decrypted
- Select the Secret file
- Decrypt the file : DES
  RNS

- Transaction
- View the transaction of logged user
- Change password.

Encryption Process :
Normal File Encrypt using RNS
F1 (encrypted by RNS)
F1 (encrypted by RNS)
Encrypt using DES F2
(Encrypted by DES and RNS)
Decryption Process :
F2 Decrypt using DES
F1(encrypted by RNS)
F1 encrypted by RNS)
Decrypt with RNS Normal File

Identity Token Generation Name : username
DOJ : 12-07-2013
Email ID: user@gmail.com
Domain : Doctor
Sub-Domain : Ortho

Hashing :
username +12-07-2013+ user@gmail.com + Doctor +Ortho
Input : IV (128 Bit)
Output: 128 bit MD - Identity Token
Send Identity Token as a File to the Registered User ID

4.2 Use Cases

4.3 Data Flow Diagram

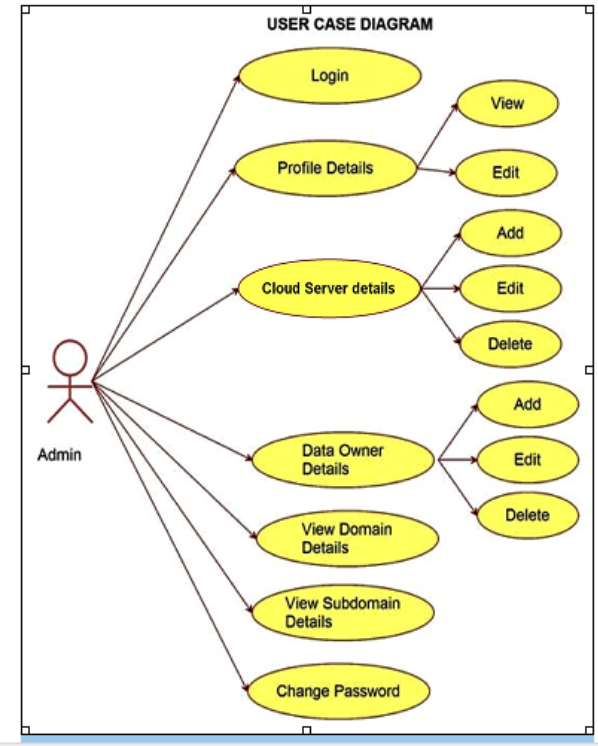

Figure 4.1: Admin

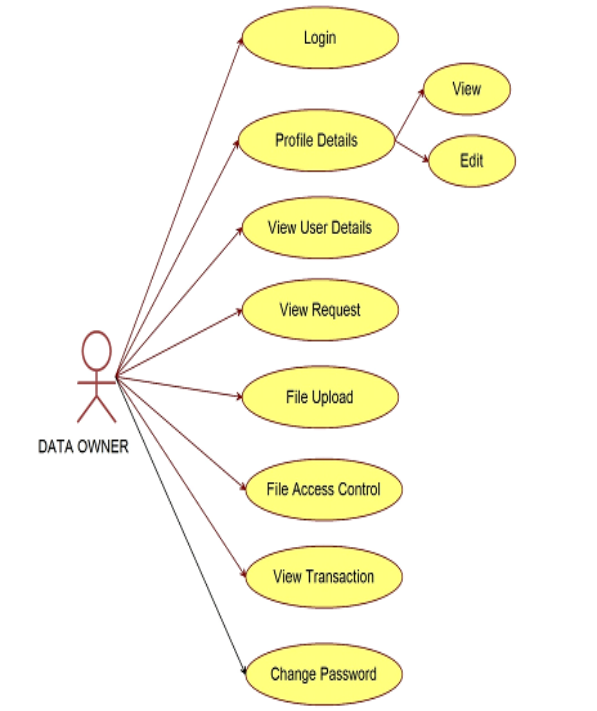

Figure 4.2: Data Owner

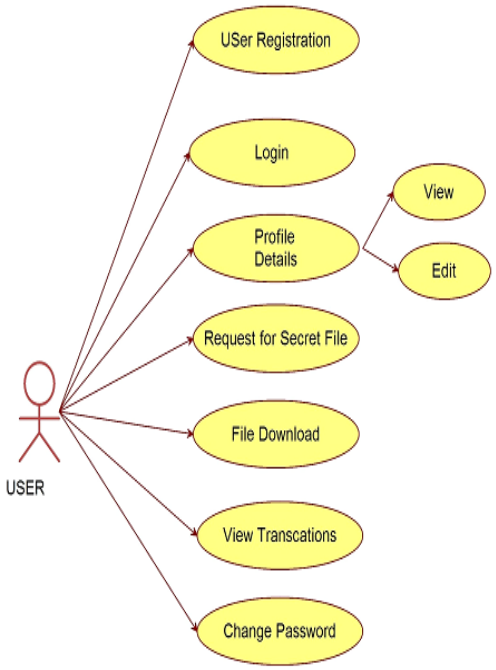

Figure 4.3: User

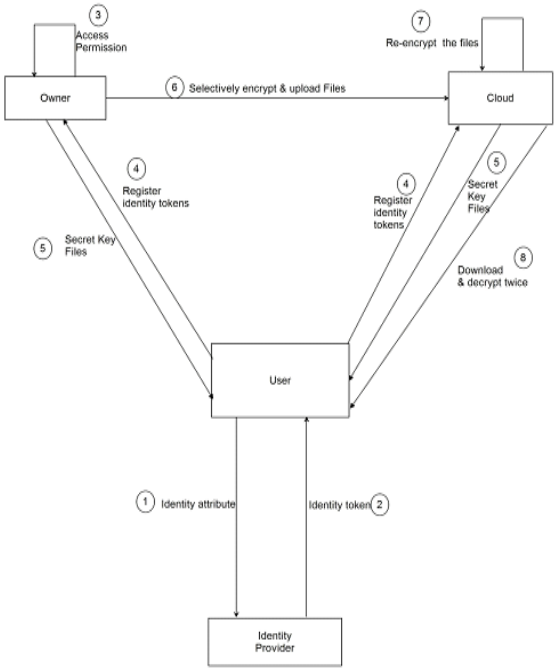

Figure 4.4: Overall

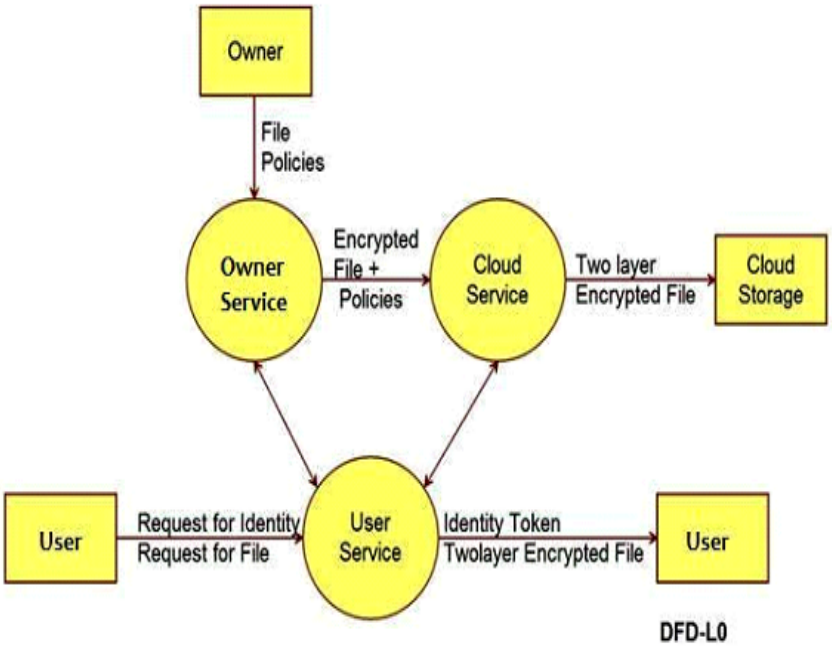

Figure 4.5: Context Analysis

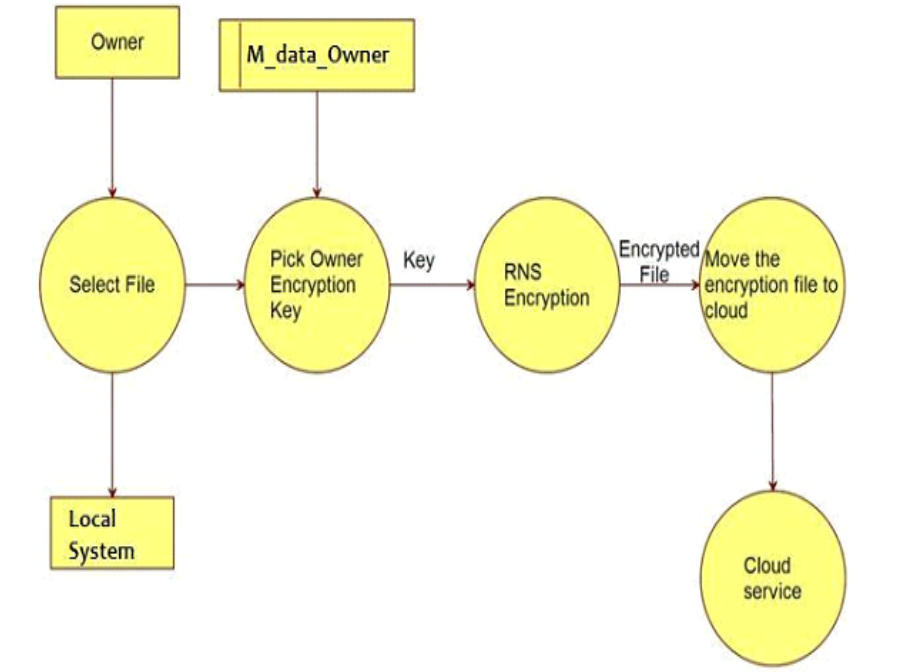

Figure 4.6: Upload Service

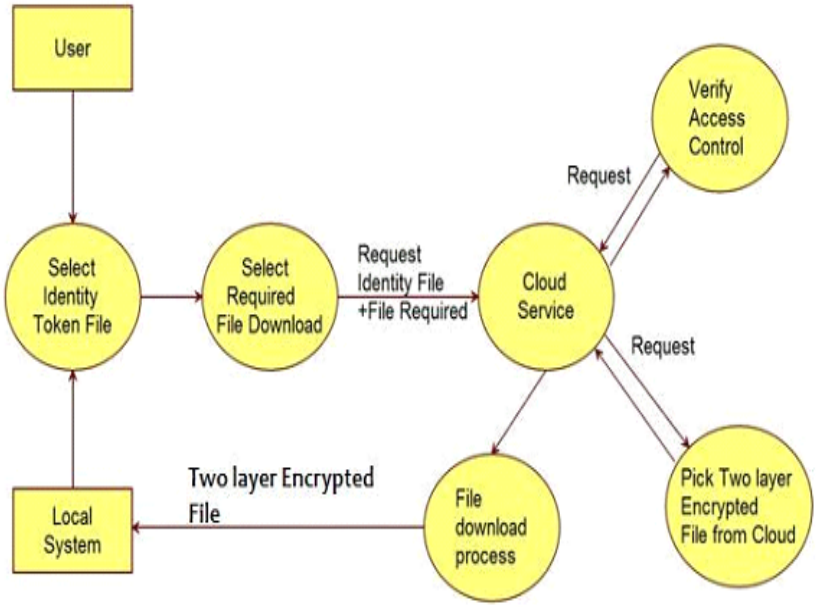

Figure 4.7: User Session File Download

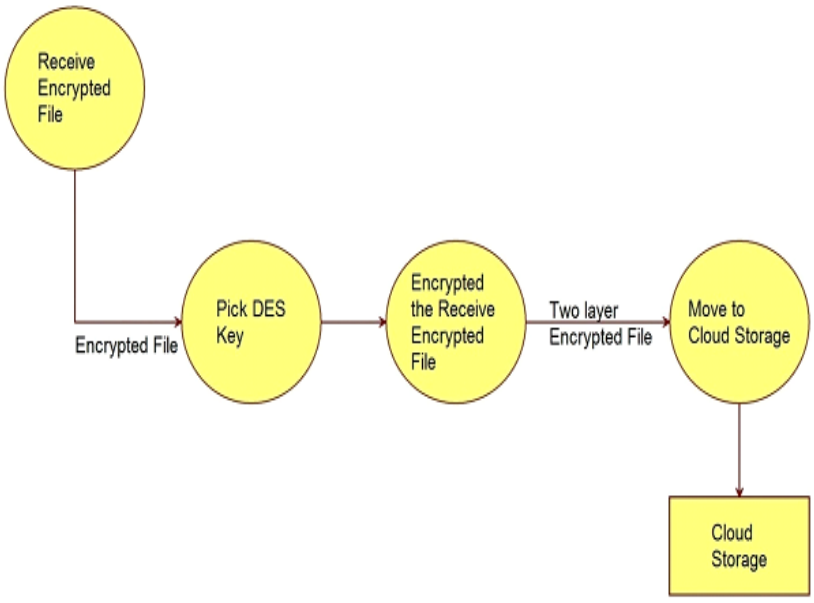

Figure 4.8: Cloud Service

Chapter 5

Theoretical Background

5.1 System Testing

Unit testing is a development procedure where programmers create tests as they develop software. The tests are simple short tests that test functionally of a particular unit or module of their code, such as a class or function.
Using open source libraries like cunit, oppunit and nun it (for C, C++) these tests can be automatically run and any problems found quickly. As the tests are developed in parallel with the source unit test demonstrates its correctness.

5.1.1 Validation and Testing

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

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

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

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

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

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

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

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

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

Integration Testing: Integration Testing can proceed in a number of different ways, which can be broadly characterized as top down or bottom up. In top down integration testing the high level control routines are tested first, possibly with the

middle level control structures present only as stubs. Subprogram stubs were presented in section2 as incomplete subprograms which are only present to allow the higher. Level control routines to be tested.

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

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

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

In the construction industry, scaffolding is a temporary, easy to assemble and disassemble, frame placed around a building to facilitate the construction of the building. The construction workers first build the scaffolding and then the building. Later the scaffolding is removed, exposing the completed building.similarly, in software testing, one particular test may need some supporting software. This software establishes can

a correct evaluation of the test take place. The scaffolding software may establish state and values for data structures as well as providing dummy external functions for the test. Different scaffolding software may be needed form one test to another test. Scaffolding software rarely is considered part of the system.

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

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

5.2 Overview

5.2.1 HTML

HTML, an initialism of Hypertext Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document by denoting certain text as headings, paragraphs, lists, and so on and to supplement that text with interactive forms, embedded images, and other objects. HTML is written in the form of labels (known as tags), surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code which can affect the behavior of web browsers and other HTML processors.

HTML is also often used to refer to content of the MIME type text/html or even more broadly as a generic term for HTML whether in its XML-descended form (such as XHTML 1.0 and later) or its form descended directly from SGML Hypertext Markup Language (HTML), the languages of the World Wide Web (WWW), allows users to produces Web pages that include text, graphics and pointer to other Web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879, SGML (Standard Generalized Markup Language), but specialized to hypertext and adapted to the Web. The idea behind Hypertext is that instead of reading text in rigid linear structure, we can easily jump from one point to another point. We can navigate through the information based on our interest and preference. A markup language is simply a series of elements, each delimited with special characters that define how text or other items enclosed within the elements should be displayed. Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

HTML can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop.

HTML provides tags (special codes) to make the document look attractive. HTML tags are not case-sensitive. Using graphics, fonts, different sizes, color, etc., can enhance the presentation of the document. Anything that is not a tag is part of the document itself.

Attributes :

The attributes of an element are name-value pairs, separated by "=", and written within the start label of an element, after the element's name. The value should be enclosed in single or double quotes, although values consisting of certain characters can be left unquoted in HTML (but not XHTML).Leaving attribute values unquoted is considered unsafe.

Most elements take any of several common attributes: id, class, style and title. Most also take language-related attributes: lang and dir.

The id attribute provides a document-wide unique identifier for an element. This can be used by style sheets to provide presentational properties, by browsers to focus attention on the specific element or by scripts to alter the contents or presentation of an element. The class attribute provides a way of classifying similar elements for presentation purposes. For example, an HTML document (or a set of documents) may use the designation class="notation" to indicate that all elements with this class value are all subordinate to the main text of the document (or documents). Such notation classes of elements might be gathered together and presented as footnotes

on a page, rather than appearing in the place where they appear in the source HTML.

An author may use the style non-attributal codes presentational properties to a particular element. It is considered better practice to use an elements son- id page and select the element with a style sheet, though sometimes this can be too cumbersome for a simple ad hoc application of styled properties. The title is used to attach sub textual explanation to an element. In most browsers this title attribute is displayed as what is often referred to as a tool tip. The generic inline span element can be used to demonstrate these various non-attributes.

The preceding displays as HTML (pointing the cursor at the abbreviation should display the title text in most browsers).

Advantages :
A HTML document is small and hence easy to send over the net.
It is small because it does not include formatted information.
HTML is platform independent.
HTML tags are not case-sensitive.

5.3 JavaScript

JavaScript is a script-based programming language that was developed by Netscape Communication Corporation. JavaScript was originally called Live Script and renamed as JavaScript to indicate its relationship with Java. JavaScript supports the development of both client and server components of Web-based applications. On the client side, it can be used to write programs that are executed by a Web browser within the context of a Web page. On the server side, it can be used to write Web server programs that can process information submitted by a Web browser and then update the browsers display accordingly.

Even though JavaScript supports both client and server Web programming, we prefer JavaScript at Client side programming since most of the browsers supports it. JavaScript is almost as easy to learn as HTML, and JavaScript statements can be included in HTML documents by enclosing the statements between a pair of scripting tags .

Here are a few things we can do with JavaScript :
Validate the contents of a form and make calculations.

Add scrolling or changing messages to the Browsers status line.
Animate images or rotate images that change when we move the mouse over them.
Detect the browser in use and display different content for different browsers.
Detect installed plug-ins and notify the user if a plug-in is required.
We can do much more with JavaScript, including creating entire application.

JavaScript and Java are entirely different languages. A few of the most glaring differences are:
Java applets are generally displayed in a box within the web document; JavaScript can affect any part of the Web document itself.
While JavaScript is best suited to simple applications and adding interactive features to Web pages; Java can be used for incredibly complex applications.

There are many other differences but the important thing to remember is that JavaScript and Java are separate languages. They are both useful for different things; in fact they can be used together to combine their advantages.

JavaScript can be used for Sever-side and Client-side scripting.
It is more flexible than VBScript.
JavaScript is the default scripting languages at Client-side since all the browsers supports it.

Java Technology :

Initially the language was called as oak but it was renamed as Java in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

Java is a programmers language.
Java is cohesive and consistent.
Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

Finally, Java is to Internet programming where C was to system programming.

Importance of java to internet :
Java has had a profound effect on the Internet. This is because; Java expands the

Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.
Java can be used to create two types of programs :
Applications and Applets: An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++. Javas ability to create Applets makes it important. An Applet is an application designed to be transmitted over the Internet and executed by a Java compatible web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can react to the user input and dynamically change.

Features of Java Security :
Every time you that you download a normal program you are risking a viral infection. Prior to java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a firewall between a network application and your computer.

When you use a java-compatible web browser, you can safely download java applets without fear of virus infection or malicious intent.

Portability :

For programs to be dynamically downloaded to all the various types of platforms connected to the internet, some means of generating portable executable code is needed .as you will see, the same mechanism that helps ensure security also helps create portability. Indeed, javas solution to these two problems is both elegant and efficient. THE BYTE CODE :
The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the

Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just In Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece by piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

Java Virtual Machine (JVM)

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code thats has been generated by the compiler will not corrupt the machine that its loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

Overall Description :

Java programming uses to produce byte codes and executes them. The first box indicates that the Java source code is located in a. Java file that is processed with a Java compiler called javac. The Java compiler produces a file called a. class file, which contains the byte code. The .Class file is then loaded across the network or loaded locally on your machine into the execution environment is the Java virtual machine, which interprets and executes the byte code.

Java Architecture

Java architecture provides a portable, robust, high performing environment for development. Java provides portability by compiling the byte codes for the Java Virtual Machine, which is then interpreted on each platform by the run-time environment.

Java is a dynamic system, able to load code when needed from a machine in the same room or across the planet.

Compilation of code

When you compile the code, the Java compiler creates machine code (called byte code) for a hypothetical machine called Java Virtual Machine (JVM). The JVM is supposed to execute the byte code. The JVM is created for overcoming the issue of portability. The code is written and compiled for one machine and interpreted on all machines. This machine is called Java Virtual Machine.

Introduction to Servlets:

Servlets provide a Java(TM)-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions, platform-specific APIs, and incomplete interfaces.

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side – object byte codes that can be dynamically loaded off the net. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform-independent, dynamically loadable, plug gable helper byte code objects on the server side that can be used to dynamically extend server-side functionality.

What is a **Servlet**?

Servlets are modules that extend request/response-oriented servers, such as Java-enabled web servers. For example, a servlet might be responsible for taking data in an HTML order-entry form and applying the business logic used to update a companys order database.

Servlets are to servers what applets are to browsers. Unlike applets, however, Servlets have no graphical user interface. Servlets can be embedded in many different servers because the servlet API, which you use to write Servlets, assumes nothing about the server's environment or protocol. Servlets have become most widely used within HTTP servers; many web servers support the Servlet API.

Use Servlets instead of CGI Scripts

Servlets are an effective replacement for CGI scripts. They provide a way to generate dynamic documents that is both easier to write and faster to run. Servlets also address the problem of doing server-side programming with platform-specific APIs: they are developed with the Java Servlet API, a standard Java extension.

So use Servlets to handle HTTP client requests. For example, have Servlets process data posted over HTTPS using an HTML form, including purchase order or credit card data. A servlet like this could be part of an order-entry and processing system, working with product and inventory databases, and perhaps an on-line payment system.

Other Uses for Servlets

Here are a few more of the many applications for Servlets

Allowing collaboration between people. A Servlet can handle multiple requests concurrently, and can synchronize requests. This allows Servlets to support systems such as on-line conferencing.

Forwarding requests. Servlets can forward requests to other servers and Servlets. Thus Servlets can be used to balance load among several servers that mirror the same content, and to partition a single logical service over several servers, according to task type or organizational boundaries.

Architecture of the Servlet Package

The javax.servlet package provides interfaces and classes for writing Servlets. The architecture of the package is described below.

The Servlet Interface

The central abstraction in the Servlet API is the Servlet interface. All Servlets implement this interface, either directly or, more commonly, by extending a class that implements it such as HttpServlet.

The Servlet interface declares, but does not implement, methods that manage the servlet and its communications with clients. Servlet writers provide some or all of these methods when developing a servlet.

Client Interaction

When a servlet accepts a call from a client, it receives two objects:

A ServletRequest, which encapsulates the communication from the client to the server.

A ServletResponse, which encapsulates the communication from the servlet back to the client.

ServletRequest and ServletResponse are interfaces defined by the javax.servlet package.

The ServletRequest Interface

The ServletRequest interface allows the servlet access to: Information such as the

names of the parameters passed in by the client, the protocol (scheme) being used by the client, and the names of the remote host that made the request and the server that receive the input stream, ServletInputStream. Servlets use the input stream to get data from clients that use application protocols such as the HTTP POST and PUT methods.

Interfaces that extend ServletRequest interface allow the servlet to retrieve more protocol-specific data. For example, the HttpServletRequest interface contains methods for accessing HTTP-specific header information.

The ServletResponse Interface

The ServletResponse interface gives the servlet methods for replying to the client. It allows the servlet to set the content length and MIME type of the reply.

Provides an output stream, ServletOutputStream, and a Writer through which the servlet can send the reply data.

Interfaces that extend the ServletResponse interface give the servlet more protocol-specific capabilities. For example, the HttpServletResponse interface contains methods that allow the servlet to manipulate HTTP-specific header information.

Additional Capabilities of HTTP Servlets

The classes and interfaces described above make up a basic Servlet. HTTP Servlets have some additional objects that provide session-tracking capabilities. The servlet writer can use these APIs to maintain state between the servlet and the client that persists across multiple connections during some time period. HTTP Servlets also have objects that provide cookies. The servlet writer uses the cookie API to save data with the client and to retrieve this data.

The classes mentioned in the Architecture of the Servlet Package section are shown in the example in bold:

SimpleServlet extends the HttpServlet class, which implements the Servlet interface.
SimpleServlet overrides the doGet method in the HttpServlet class. The doGet method is called when a client makes a GET request (the default HTTP request method) and results in the simple HTML page being returned to the client.
Within the doGet method, An HttpServletRequest object represents the users request.

An HttpServletResponse object represents the response to the user.
Because text data is returned to the client, the reply is sent using the Writer object obtained from the HttpServletResponse object.

Server Life Cycle

Each servlet has the same life cycle
A server loads and initializes the servlet
The servlet handles zero or more client requests
The server removes the servlet

Initializing a Servlet

When a server loads a servlet, the server runs the servlet's init method. Initialization completes before client requests are handled and before the servlet is destroyed.

Even though most Servlets are run in multi-threaded servers, Servlets have no concurrency issues during servlet initialization. The server calls the init method once, when the server loads the servlet, and will not call the init method again unless the server is reloading the servlet. The server cannot reload a servlet until after the server has destroyed the servlet by calling the destroy method.

The init Method:

The init method provided by the HttpServlet class initializes the servlet and logs the initialization. To do initialization specific to your servlet, override the init () method following these rules:

If an initialization error occurs that renders the servlet incapable of handling client requests, throw an Unavailable Exception.

Initialization Parameters:

The second version of the init method calls the getInitParameter method. This method takes the parameter name as an argument and returns a String representation of the parameter's value.

The specification of initialization parameters is server-specific. In the Java Web Server, the parameters are specified with a servlet is added then configured in the Administration Tool. For an explanation of the Administration screen where this setup

is performed, see the Administration Tool: Adding Servlets online help document.

In some cases, if we need to get the parameter names, we can use the getParameterNames method.

Destroying a Servlet

Servlets run until the server is destroys them, for example at the request of a system administrator. When a server destroys a servlet, the server runs the servlet's destroy method. The method is run once; the server will not run that servlet again until after the server reloads and reinitializes the servlet.

When the destroy method runs, another thread might be running a service request. The Handling Service Threads at Servlet Termination section shows you how to provide a clean shutdown when there could be long-running threads still running service requests.

Using the Destroy Method

The destroy method provided by the HttpServlet class destroys the servlet and logs the destruction. To destroy any resources specific to your servlet, override the destroy method. The destroy method should undo any initialization work and synchronize persistent state with the current in-memory state.

A server calls the destroy method after all service calls have been completed, or a server-specific number of seconds have passed, whichever comes first. If your servlet handles any long-running operations, service methods might still be running when the server calls the destroy method. You are responsible for making sure those threads complete. The next section shows you how.

The destroy method shown above expects all client interactions to be completed when the destroy method is called, because the servlet has no long-running operations.

Java Server Pages

Java Server Pages technology lets you put snippets of servlet code directly into a text-based document. A JSP page is a text-based document that contains two types of text: static template data, which can be expressed in any text-based format such as HTML, WML, and XML, and JSP elements, which determine how the page constructs dynamic content.

Java Server Page (JSP): An extensible Web technology that uses template data, custom elements, scripting languages, and server-side Java objects to return dynamic content to a client. Typically the template data is HTML or XML elements, and in many cases the client is a Web browser.

Java Server Pages (JSP) lets you separate the dynamic part of your pages from the static HTML. You simply write the regular HTML in the normal manner, using whatever Web-page-building tools you normally use. You then enclose the code for the dynamic parts in special tags, most of which starts.

You normally give your file a .jsp extension, and typically install it in any place you could place a normal Web page. Although what you write often looks more like a regular HTML file than a servlet, behind the scenes, the JSP page just gets converted to a normal servlet, with the static HTML simply being printed to the output stream associated with the servlet's service method. This is normally done the first time the page is requested, and developers can simply request the page themselves when first installing it if they want to be sure that the first real user doesn't get a momentary delay when the JSP page is translated to a servlet and the servlet is compiled and loaded. Note also that many Web servers let you define aliases that so that a URL that appears to reference an HTML file really points to a servlet or JSP page.

Aside from the regular HTML, there are three main types of JSP constructs that you embed in a page: scripting elements, directives, and actions. Scripting elements let you specify Java code that will become part of the resultant servlet, directives let you control the overall structure of the servlet, and actions let you specify existing components that should be used, and otherwise control the behavior of the JSP engine. To simplify the scripting elements, you have access to a number of predefined variables such as request in the snippet above.

J2EE Platform Overview

The J2EE platform is designed to provide server-side and client-side support for developing distributed, multi-tier applications. Such applications are typically configured as a client tier to provide the user interface, one or more middle-tier modules that provide client services and business logic for an application, and back-end enterprise information systems providing data management.

Multitier Model

The J2EE platform provides a multi-tier distributed application model. This means

that the various parts of an application can run on different devices. The J2EE architecture defines a client tier, a middle tier (consisting of one or more sub-tier), and a back-end tier. The client tier supports a variety of client types, both outside and inside of corporate firewalls. The middle tier supports client services through Web containers in the Web tier and supports business logic component services through JavaBeans TM. On the back end, the enterprise information systems in the tier are accessible by way of standard APIs.

Container-Based Component Management

Central to the J2EE component-based development model is the notion of containers. Containers are standardized runtime environments that provide specific services to components. Components can expect these services to be available on any J2EE platform from any vendor. For example, all J2EE Web containers provide runtime support for responding to client requests, performing request-time processing (such as invoking JSP pages or servlet behavior), and returning results to the client. In addition, they provide APIs to support user session management. All WEB containers provide automated support for transaction and life cycle management of WEB components, as well as bean lookup and other services. Containers also provide standardized access to enterprise information systems; for example, providing access to relational data through the JDBC API. In addition, containers provide a mechanism for selecting application behaviors at assembly or deployment time. Through the use of deployment descriptors (XML files that specify component and container behavior), components can be configured to a specific containers environment when deployed, rather than in component code. Features that can be configured at deployment time include security checks, transaction control, and other management responsibilities. While the J2EE specification defines the component containers that a platform implementation must support, it doesnt specify or restrict the containers configurations. Thus, both container types can run on a single platform, Web containers can live on one platform and WEB containers on another or a J2EE platform can be made up of multiple containers on multiple platforms.

Support for Client Components

The J2EE client tier provides support for a variety of client types, both within the enterprise firewall and outside. Clients can be offered through Web browsers by using plain HTML pages, HTML generated dynamically by Java Server PagesTM.

Support for Business Logic Components

While simple J2EE applications may be built largely in the client tier, business logic

is often implemented on the J2EE platform in the middle tier as Java Beans components (also known as enterprise beans). Enterprise beans allow the component or application developer to concentrate on the business logic while the complexities of delivering a reliable, scalable service are handled by the WEB container.

In many ways, the J2EE platform and Java Beans architecture have complementary goals. The Java Beans component model is the backbone of industrial-strength application architectures in the J2EE programming model. The J2EE platform complements the specification by:

Fully specifying the APIs that an enterprise bean developer can use to implement enterprise beans

Defining the larger, distributed programming environment in which enterprise beans are used as business logic components

J2EE Platform Benefits

With features designed to expedite the process of developing distributed applications, the J2EE platform offers several benefits:

Simplified architecture and development
Freedom of choice in servers, tools, and components
Integration with existing information systems
Scalability to meet demand variations
Flexible security model

Simplified Architecture and Development

The J2EE platform supports a simplified, component-based development model. Because it is based on the Java programming language and the Java 2 Platform, Standard Edition (J2SETM platform), this model offers Write-Once-Run-Anywhere portability, supported by any server product that conforms to the J2EE standard.

The component-based J2EE development model can enhance application development productivity in a number of ways:

Maps easily to application functionalityComponent-based application models map easily and flexibly to the functionality desired from an application. As the examples presented throughout this book illustrate, the J2EE platform provides a variety of ways to configure the architecture of an application, depending on such things as

client types required, level of access required to data sources, and other considerations. Component-based design also simplifies application maintenance, since components can be updated and replaced independentlynew functionality can be shimmed into existing applications simply by updating selected components.

Enables assembly- and deploy-time behaviorsBecause of the high level of service standardization, much of the code of a J2EE application can be generated automatically by tools, with minimal developer intervention. In addition, components can expect standard services to be available in the runtime environment and can dynamically connect to other components by means of consistent interfaces. As a result, many application behaviors can be configured at application assembly or deployment time, without recoding. Component developers can communicate requirements to application deployers through specific deployment descriptors and settings. Tools can automate this process to further expedite development.

Supports division of laborComponents help divide the labor of application development among specific skill sets, enabling each member of a development team to focus on his or her ability. Web page authors can create JSP templates, Java programming language coders can implement application behavior, domain experts can develop business logic, and application developers and integrators can assemble and deploy applications. This division of labor also expedites application maintenance. For example, the user interface is the most dynamic part of many applications, particularly on the Web. With the J2EE platform, Web page authors can tweak the look and feel of JSP pages without programmer intervention. The J2EE specifications define a number of roles, including application component provider, application assembler, and application deployer. On some development teams, one or two people may perform all these roles, while on others.

Integrating Existing Enterprise Information Systems

The J2EE platform, together with the J2SE platform, includes a number of industries standard APIs for accessing existing enterprise information systems. Basic access to these systems is provided by the following APIs:
The J2EE Connector architecture is the infrastructure for interacting with a variety of Enterprise Information System types, including ERP, CRM, and other legacy systems.

The JDBCTM API is used for accessing relational data from the Java programming language.

The Java Transaction API (JTA) is the API for managing and coordinating transactions across heterogeneous enterprise information systems.

The Java Naming and Directory Interface TM (JNDI) is the API for accessing information in enterprise name and directory services.

The Java Message Service (JMS) is the API for sending and receiving messages via enterprise messaging systems such as IBM MQ Series and TIBCO Rendezvous. In the J2EE platform version 1.3, message-driven beans provide a component-based approach to encapsulating messaging functionality.

Java APIs for XML provide support for integration with legacy systems and applications, and for implementing Web services in the J2EE platform. In addition, specialized access to enterprise resource planning and mainframe systems such as IBMs CICS and IMS is provided through the J2EE Connector architecture. Since each of these systems is highly complex and specialized, they require unique tools and support to ensure utmost simplicity to application developers.

Choice of Servers, Tools, and Components

The J2EE standard and J2EE brand have created a huge marketplace for servers, tools, and components. The J2EE brand on a server product ensures the consistent level of service that is fundamental to the goals of the J2EE platform. At the same time, J2EE standards ensure a lively marketplace for tools and components. Based on past experience and industry momentum, all leading enterprise software vendors are expected to provide the marketplace for J2EE 1.3 products. The standardization and branding of the J2EE platform provides many benefits, including:

A range of server choices - Application development organizations can expect J2EE branded platforms from a variety of vendors, providing a range of choices in hardware platforms, operating systems, and server configurations. This ensures that businesses get a choice of servers appropriate to their needs.

Designed for tool support - Both enterprise beans and JSP page components are designed to be manipulated by graphical development tools and to allow automating many of the application development tasks traditionally requiring the ability to write and debug code. Both J2EE server providers and third-party tool developers have developed tools that conform to J2EE standards and support various application

development tasks and styles. Application developers have a choice of tools to manipulate and assemble components, and individual team members may choose tools that best suit their specific requirements. A marketplace for components - Component-based design ensures that many types of behavior can be standardized, packaged, and reused by any J2EE application. Component vendors will provide a variety of off-the-shelf component solutions, including accounting beans, user interface templates, and even vertical market functionality of interest in specific industries. Application architects get a choice of standardized components to handle common or specialized tasks. The J2EE standard and associated branding programs ensure that solutions are compatible. By setting the stage for freedom of choice, the J2EE platform makes it possible to develop with confidence that the value of your investment will be protected.

Scales Easily

J2EE containers provide a mechanism that supports simplified scaling of distributed applications, with no application development effort. Because J2EE containers provide components with transaction support, database connections, life cycle management, and other features that influence performance, they can be designed to provide scalability in these areas. For example, containers may pool database connections, providing clients with quick, efficient access to data. Because containers may run on multiple systems, Web containers can automatically balance load in response to fluctuating demand.

Simplified, Unified Security Model

The J2EE security model is designed to support single sign on access to application services. Component developers can specify the security requirements of a component at the method level to ensure that only users with appropriate permissions can access specific data operations. While both Java Beans technology and Java Servlet APIs provide programmatic security control, the basic role-based security mechanism (where groups of users share specific permissions) is specified entirely at application deployment time. This provides both greater flexibility and better security control.

J2EE Application Scenarios

The J2EE specifications encourage architectural diversity. The J2EE specifications and technologies make few assumptions about the details of API implementations. The application-level decisions and choices are ultimately a trade-off between functional richness and complexity. The J2EE programming model is flexible enough for applications that support a variety of client types, with both the Web container

and WEB container as optional.

The following enterprise requirements heavily influenced the choices made in developing the sample application:

The need to make rapid and frequent changes to the look and feel of the application.
The need to partition the application along the lines of presentation and business logic so as to increase modularity.
The need to simplify the process of assigning suitably trained human resources to accomplish the development task such that work can proceed along relatively independent but cooperating tracks.

The need to have developers familiar with back-office applications unburdened from GUI and graphic design work, for which they may not be ideally qualified.

The need to have the necessary vocabulary to communicate the business logic to teams concerned with human factors and the aesthetics of the application.

The ability to assemble back-office applications using components from a variety of sources, including off-the-shelf business logic components .

The ability to deploy transactional components across multiple hardware and software platforms independently of the underlying database technology. The ability to externalize internal data without having to make many assumptions about the consumer of the data and to accomplish this in a loosely coupled manner Clearly, relaxing any or all of these requirements would influence some of the application-level decisions and choices that a designer would make. Although it is reasonable to speak of throw-away presentation logic (that is, applications with a look and feel that ages rapidly), there is still significant inertia associated with business logic. This is even truer in the case of database schemas and data in general. It is fair to say that as one moves further away from EIS resources, the volatility of the application code increases dramatically; that is, the codes shelf life drops significantly.

Multitier Application Scenario
JSP pages, supported by Servlets, generate dynamic Web content for delivery to the client. The Web container hosts application components that use EIS resources to service requests from Web-tier components. This architecture decouples data access

from the applications user interface. The architecture is also implicitly scalable. Application back-office functionality is relatively isolated from the end-user look and feel.

It is worth noting that XML plays an integral role in this scenario. The ability to both produce and consume XML data messages in the Web container is an extremely flexible way to embrace a diverse set of client types. These platforms range from general purpose XML-enabled browsers to specialized XML rendering engines targeting vertical solutions. XML data messages typically use HTTP as their transport protocol. Java and XML are complementary technologies: The Java language offers portable code, XML provides portable data. In the Web tier, the question of whether to use JSP pages or Servlets comes up repeatedly. JSP technology is intended for application user interface components, while Java Servlets are preferred for request processing and application control logic. Servlets and JSP pages work together to provide dynamic content from the Web tier.

The stand-alone client may be one of three types:

Stand-alone clients, implemented in the Java language or another programming language, consuming dynamic Web content (usually XML data messages). In this scenario, the Web container essentially handles XML transformations and provides Web connectivity to clients. Presentation logic occurs in the client tier. The Web tier handles business logic and may directly access EIS resources. Ideally, business logic is implemented as enterprise beans to take advantage of the rich enterprise beans component model.

Stand-alone Java application clients accessing enterprise information system resources directly using JDBC or Connectors. In this scenario, presentation and business logic are co-located on the client platform and may in fact be tightly integrated into a single application. This scenario is classic two-tier client-server architecture, with its associated distribution, maintenance, and scalability issues.

There are a number of scenarios in which the use of enterprise beans in an application would be considered overkill: sort of like using a sledgehammer to crack a nut. The J2EE specification doesnt mandate a specific application configuration, nor could it realistically do so. The J2EE platform is flexible enough to support the application configuration most appropriate to a specific application design requirement.

The Web container hosts both presentation and business logic, and it is assumed that JDBC and the J2EE Connector architecture are used to access EIS resources.

In many cases, J2EE platform providers may co-locate their Web and WEB containers, running them within the same Java Virtual Machine (JVM). J2EE applications deployed on such an implementation are still considered Multitier applications, because of the division of responsibilities that the separate technologies imply.

The peer-level communications between WEB containers is currently a more tightly coupled solution most suitable for intranet environments. With support for JMS and message-driven beans, the J2EE 1.3 platform makes developing loosely coupled intranet solutions increasingly practical.

Future releases of the J2EE platform will provide additional functionality in the form of Java APIs for XML, which enable more complete support for loosely coupled applications through XML-based Web services.

Communication Technologies

Communication technologies provide mechanisms for communication between clients and servers and between collaborating objects hosted by different servers. The J2EE specification requires support for the following types of communication technologies:
Remote method invocation protocols
Object Management Group protocols
Messaging technologies
Data formats

Internet protocols define the standards by which the different pieces of the J2EE platform communicate with each other and with remote entities. The J2EE platform supports the following Internet protocols:

TCP/IPTransport Control Protocol over Internet Protocol. These two protocols provide for the reliable delivery of streams of data from one host to another. Internet Protocol (IP), the basic protocol of the Internet, enables the unreliable delivery of individual packets from one host to another. IP makes no guarantees as to whether the packet will be delivered, how long it will take, or if multiple packets will arrive in the order they were sent. The Transport Control Protocol (TCP) adds the notions of connection and reliability.
HTTP 1.0Hypertext Transfer Protocol. The Internet protocol used to fetch hypertext objects from remote hosts. HTTP messages consist of requests from client to server and responses from server to client.

SSL 3.0Secure Socket Layer. A security protocol that provides privacy over the Internet. The protocol allows client-server applications to communicate in a way that cannot be eavesdropped or tampered with. Servers are always authenticated and clients are optionally authenticated.

Remote Method Invocation Protocols

Remote Method Invocation (RMI) is a set of APIs that allow developers to build distributed applications in the Java programming language. RMI uses Java language interfaces to define remote objects and a combination of Java serialization technology and the Java Remote Method Protocol (JRMP) to turn local method invocations into remote method invocations. The J2EE platform supports the JRMP protocol, the transport mechanism for communication between objects in the Java language in different address spaces.

Object Management Group Protocols

Object Management Group (OMG) protocols allow objects hosted by the J2EE platform to access remote objects developed using the OMGs Common Object Request Broker Architecture (CORBA) technologies and vice versa. CORBA objects are defined using the Interface Definition Language (IDL). An application component provider defines the interface of a remote object in IDL and then uses an IDL compiler to generate client and server stubs that connect object implementations to an Object Request Broker (ORB), a library that enables CORBA objects to locate and communicate with one another. ORBs communicate with each other using the Internet Inter-ORB Protocol (IIOP). The OMG technologies required by the J2EE platform are Java IDL and RMI-IIOP.

Java IDL

Java IDL allows Java clients to invoke operations on CORBA objects that have been defined using IDL and implemented in any language with a CORBA mapping. Java IDL is part of the J2SE platform. It consists of a CORBA API and ORB. An application component provider uses the idlj IDL compiler to generate a Java client stub for a CORBA object defined in IDL. The Java client is linked with the stub and uses the CORBA API to access the CORBA object.

RMI-IIOP

RMI-IIOP is an implementation of the RMI API over IIOP. RMI-IIOP allows application component providers to write remote interfaces in the Java programming language. The remote interface can be converted to IDL and implemented in any

other language that is supported by an OMG mapping and an ORB for that language. Clients and servers can be written in any language using IDL derived from the RMI interfaces. When remote interfaces are defined as Java RMI interfaces, RMI over IIOP provides interoperability with CORBA objects implemented in any language.

RMI-IIOP contains:
The rmic compiler, which generates: - Client and server stubs that work with any ORB. An IDL file compatible with the RMI interface. To create a C++ server object, an application component provider would use an IDL compiler to produce the server stub and skeleton for the server object.
A CORBA API and ORB. Application clients must use RMI-IIOP to communicate with enterprise beans.

5.4 MySQL

MySql is a relational database management system, which organizes data in the form of tables. MySQL is one of many databases servers based on RDBMS model, which manages a seer of data that attends three specific things-data structures, data integrity and data manipulation. With MySQL cooperative server technology we can realize the benefits of open, relational systems for all the applications. MySQL makes efficient use of all systems resources, on all hardware architecture; to deliver unmatched performance, price performance and scaleability.Any DBMS to be called as RDBMS has to satisfy Dr.E.F.Codds rules.

Distinct Features of MySql:

MYSQL IS PORTABLE:

The MySQL RDBMS is available on wide range of platforms ranging from PCs to super computers and as a multi user loadable module for Novel NetWare, if you develop application on system you can run the same application on other systems without any modifications.

MYSQL IS COMPATIBLE:

MySQL commands can be used for communicating with IBM DB2 mainframe RDBMS that is different from MySQL , that is MySQL compatible with DB2 .MySQL RDBMS is a high performance fault tolerant DBMS , which is specially designed for

online transaction processing and for handling large database applications.

MULTITHREADED SERVER ARCHITECTURE:

MySQL adaptable multithreaded server architecture delivers scalable high performance for very large number of users on all hardware architecture including symmetric multiprocessors (sumps) and loosely coupled multiprocessors. Performance is achieved by eliminating CPU, I/O, memory and operating system bottlenecks and by optimizing the MySQL DBMS server code to eliminate all internal bottlenecks.

Features of MySQL

Most popular RDBMS in the market because of its ease of use
Client/server architecture.
Data independence.
Ensuring data integrity and data security.
Managing data concurrency.
Parallel processing support for speed up data entry and online transaction processing used for applications.
DB procedures, functions and packages.

Dr.E.F.OCDDs RULES

These rules are used for valuating a product to be called as relational database management systems. Out of 12 rules, a RDBMS product should satisfy at least 8 rules +rule called rule 0 that must be satisfied.
RULE 0: FOUNDATION RULE:
For any system that is to be advertised as ,or claimed to be relational DBMS. That system should manage database with in self, with out using an external language.

RULE 1.INFORMATION RULE
All information in relational database is represented at logical level in only one way as values in tables.

RULE 2.GUARANTEED ACCESS:

Each and every data in a relational database is guaranteed to be logically accessibility by using to a combination of table name, primary key value and column name.

RULE 3. SYSTEMATIC TREATMENT OF NULL VALUES

Null values are supported for representing missing information and inapplicable information. They must be handled in systematic way, independent of data types.

RULE 4. DYNAMIC ONLINE CATALOG BASED RELATION MODEL:

The database description is represented at the logical level in the same way as ordinary data so that authorized users can apply the same relational language to its interrogation as they do to the regular data.

RULE 5: COMPRHENSIVE DATA SUB LANGUAGE

A relational system may support several languages and various models of terminal use. However there must be one language whose statement can express all of the following:

Data Definitions, View Definitions, Data Manipulations, Integrity, Constraints, Authorization and transaction boundaries.

RULE 6: VIEW UPDATING
Any view that is theoretically that updatable if changes can be made to the tables that effect the desired changes in the view.
RULE 7.HIGH LEVEL UPDATE, INSERT and DELETE
The capability of handling a base relational or derived relational as a single operand applies not only retrieval of data also to its insertion, updating, and deletion.

RULE 8.PHYSICAL DATA INDEPENDENCE

Application program and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access method.

RULE 9.LOGICAL DATA INDEPENDENCE

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage repres3entation or access methods.

RULE 10: INTEGRITY INDEPENDENCE:
Integrity constraints specific to particular database must be definable in the relational

data stored in the catalog, not in application program.

RULE 11: DISTRIBUTED INDEPENDENCE:
Weather or not a system support data base distribution, it must have a data sub-language that can support distributed databases without changing the application program.

RULE 12: NON SUB-VERSION:
If a relational system has low level language, that low language cannot use to sub-version or by pass the integrity rules and constraints expressed in the higher level relational language.

MYSQL SUPPORTS THE FOLLOWING CODDS RULES:
Rule 1: Information Rule (Representation of information)-YES.
Rule 2: Guaranteed Access-YES.
Rule 3: Systematic treatment of Null values-YES.
Rule 4: Dynamic on-line catalog-based Relational Model-YES.
Rule 5: Comprehensive data sub language-YES.
Rule 6: View Updating-PARTIAL.
Rule 7: High-level Update, Insert and Delete-YES.
Rule 8: Physical data Independence-PARTIAL.
Rule 9: Logical data Independence-PARTIAL.
Rule 10: Integrity Independence-PARTIAL.
Rule 11: Distributed Independence-YES.
Rule 12: Non-subversion-YES.

Chapter 6

Implementation

6.1 Admin Login

```
<!DOCTYPE html>
<html >
<%!
public int convert(String str)
{
        int conv=0;
        if(str==null)
        {
                str="0";
        }
        else if((str.trim()).equals("null"))
        {
                str="0";
        }
        else if(str.equals(""))
        {
                str="0";
        }
        try
        {
                conv=Integer.parseInt(str);
        }
        catch(Exception e)
        {
        }
        return conv;
```

```
}
%>
<head>
 <meta charset="UTF-8">
 <title>Flat Login Form 3.0</title>

 <link rel="stylesheet" href="https://cdnjs.cloudflare
 .com/ajax/libs/meyer-reset/2.0/reset.min.css">

  <link rel='stylesheet prefetch' href='http://fonts.
  googleapis.com/css?family=Roboto:400,100,300,500,700,
  900|RobotoDraft:400,100,300,500,700,900'>
<link rel='stylesheet prefetch' href='http://maxcdn.bo
otstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.css'>

      <link rel="stylesheet" href="CSS/style.css">


</head>

<body>

<div     style="position: centre;  left: -10px;" >


        <img src="<%=request.getContextPath() %>/AllFiles/Imag
        es/head_Java copy.png" width="100%" height="120px" ></img>
</div>




<!-- Form Mixin-->
<!-- Input Mixin-->
<!-- Button Mixin-->
<!-- Pen Title-->
<div class="pen-title">

</div>
<!-- Form Module-->
<div class="module form-module">
```

```
  <div class="toggle"><i class="fa fa-times fa-pencil"></i>

  </div>
  <div class="form">
    <h2>Admin Login</h2>
     <form  class="login-page" name="f1" id="login" method=
     "post" action="<%=request.getContextPath()%>/AdminLogin">
      <input type="text" name="username" id="login_username"
       placeholder="UserName"/>
      <input type="password" name="password" id="login_pass
      word" placeholder="Password"/>
      <button>Login</button>
      <br>

                                  <p>
                        <a href="index1.jsp">Click
                         here </a>for Dataowner Login</p>

                                    <br>
                                  <p>
                          <a href="index2.jsp">
                         Click here </a>for User Login</p>
    </form>
  </div>


</div>
  <script src='http://cdnjs.cloudflare.com/ajax/li
  bs/jquery/2.1.3/jquery.min.js'></script>
<script src='http://codepen.io/andytran/pen/vLmRVp.
js'></script>

      <script src="js/index.js"></script>
        <%
         int no=convert(request.getParameter("no"));
         if(no==1)
         {

%>

<script type="text/javascript">
```

```
    alert('Enter Username and Password !')

    </script>


<%
        }
        if(no==2)
        {
%>
<script type="text/javascript">

    alert('Please,Enter Your Username.!')

    </script>


<%
        }
        if(no==3)
        {
%>


<script type="text/javascript">

    alert('Please,Enter Your Password.!')

    </script>


<%
        }
%>
<%
        if(no==4)
        {
%>
```

```
<script type="text/javascript">

    alert('Sorry,Invalid Username/Password!')

    </script>


<%
        }
%>
<%
        if(no==5)
        {
%>

<script type="text/javascript">

    alert('You have Logged out successfully...!')

    </script>


<%
        }
%>


</body>
</html>
```

6.2 Data Owner

```
<!DOCTYPE html>
<html >
<%!
public int convert(String str)
{
        int conv=0;
        if(str==null)
```

```
        {
                str="0";
        }
        else if((str.trim()).equals("null"))
        {
                str="0";
        }
        else if(str.equals(""))
        {
                str="0";
        }
        try
        {
                conv=Integer.parseInt(str);
        }
        catch(Exception e)
        {
        }
        return conv;
        }
%>
<head>
  <meta charset="UTF-8">
  <title>Flat Login Form 3.0</title>

  <link rel="stylesheet" href="https://cdnjs.cloudf
  lare.com/ajax/libs/meyer-reset/2.0/reset.min.css">

  <link rel='stylesheet prefetch' href='http://font
  s.googleapis.com/css?family=Roboto:400,100,300,500
  ,700,900|RobotoDraft:400,100,300,500,700,900'>
<link rel='stylesheet prefetch' href='http://maxcdn.
bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome
.min.css'>

      <link rel="stylesheet" href="CSS/style.css">


</head>
<%
        int no=convert(request.getParameter("no"));
```

```
        if(no==1)
        {

%>
                <div class="error" id="message" style="pos
                ition:absolute;top:130px;left:530px">
                        <p> Enter Username and Password !</p>
                </div>

<%
        }
        if(no==2)
        {
%>
                <div class="error" id="message" style="position:a
                bsolute;top:130px;left:530px">
                        <p> Please,Enter Your Username.!</p>
        </div>
<%
        }
        if(no==3)
        {
%>
                <div class="error" id="message" style="position:a
                bsolute;top:130px;left:530px">
                        <p> Please,Enter Your Password.!</p>
        </div>
<%
        }
%>
<%
        if(no==4)
        {
%>
                <div class="error" id="message" style="position:
                absolute;top:130px;left:530px">
                        <p> Sorry,Invalid Username/Password!</p>
        </div>
<%
        }
%>
```

```
<%
        if(no==5)
        {
%>
                <div class="success" id="message" style="position:
                absolute;top:130px;left:475px">
                        <p>You have Logged out successfully...!</p>
                </div>
<%
        }
%>

<body>
<div     style="position: centre;  left: -10px;" >


        <img src="<%=request.getContextPath() %>/AllFiles/
        Images/head_Java copy.png" width="100%" height="120px" ></img>
</div>

<!-- Form Mixin-->
<!-- Input Mixin-->
<!-- Button Mixin-->
<!-- Pen Title-->
<div class="pen-title">

</div>
<!-- Form Module-->
<div class="module form-module">
  <div class="toggle"><i class="fa fa-times fa-pencil"></i>

  </div>
  <div class="form">
    <h2>DataOwner Login</h2>
    <form  class="login-page" name="f1" id="login" method=
    "post" action="<%=request.getContextPath()%>/DataOw
    nerLogin">
      <input type="text" name="username" id="login_use
      rname" placeholder="UserName"/>
      <input type="password" name="password" id="login_
      password" placeholder="Password"/>
```

```
        <button>Login</button>
        <br>

                                        <p>
                                            <a href="index.jsp"
                    >Click here </a>for Admin Login</p>

                                          <br>
                                        <p>
                                    <a href="index2.jsp">
                            Click here </a>for User Login</p>
      </form>
    </div>


</div>
  <script src='http://cdnjs.cloudflare.com/ajax
  /libs/jquery/2.1.3/jquery.min.js'></script>
<script src='http://codepen.io/andytran/pen/
vLmRVp.js'></script>

        <script src="js/index.js"></script>

</body>
</html>
```

6.3 User Login

```
<!DOCTYPE html>
<%@page import="java.util.ArrayList"%>
<%@page import="com.dao.UserDAO"%>
<%@page import="com.dao.DAO"%>
<%@page import="java.sql.*"%>
<html >
<%!
public int convert(String str)
{
        int conv=0;
        if(str==null)
        {
```

```
                str="0";
        }
        else if((str.trim()).equals("null"))
        {
                str="0";
        }
        else if(str.equals(""))
        {
                str="0";
        }
        try
        {
                conv=Integer.parseInt(str);
        }
        catch(Exception e)
        {
        }
        return conv;
        }
%>
<head>
  <meta charset="UTF-8">
  <title>Flat Login Form 3.0</title>

  <link rel="stylesheet" href="https://cdnjs.
  cloudflare.com/ajax/libs/meyer-reset/2.0/res
  et.min.css">

  <link rel='stylesheet prefetch' href='http:/
  /fonts.googleapis.com/css?family=Roboto:400,
  100,300,500,700,900|RobotoDraft:400,100,300,
  500,700,900'>
<link rel='stylesheet prefetch' href='http://
maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css
/font-awesome.min.css'>

      <link rel="stylesheet" href="CSS/style.css">


</head>
<%
```

```
        int no=convert(request.getParameter("no"));
        if(no==1)
        {

%>
                <div class="error" id="message" style
  ="position:absolute;top:130px;left:530px">
 <p> Enter Username and Password !</p>
                </div>

<%
        }
        if(no==2)
        {
%>
                <div class="error" id="message" style=
"position:absolute;top:130px;left:530px">
    <p> Please,Enter Your Username.!</p>
        </div>
<%
        }
        if(no==3)
        {
%>
                <div class="error" id="message" style=
                "position:absolute;top:130px;left:530px">
                        <p> Please,Enter Your Password.!</p>
        </div>
<%
        }
%>
<%
        if(no==4)
        {
%>
                <div class="error" id="message" style=
        "position:absolute;top:130px;left:530px">
  <p> Sorry,Invalid Username/Password!</p>
        </div>
<%
        }
```

```
%>
<%
        if(no==5)
        {
%>
                <div class="success" id="message"
        style="position:absolute;top:130px;left:475px">
<p>You have Logged out successfully...!</p>
                </div>
<%
        }
%>
<%
 if(no==6)
        {
%>
                <div class="success" id="message"
        style="position:absolute;top:130px;left:475px">
                        <p>User Registered successfully...!</p>
                </div>
<%
        }
%>

<body>
<div    style="position: centre;  left: -10px;" >


        <img src="<%=request.getContextPath() %>/
        AllFiles/Images/head_Java copy.png" width=
        "100%" height="120px" ></img>
</div>

<!-- Form Mixin-->
<!-- Input Mixin-->
<!-- Button Mixin-->
<!-- Pen Title-->
<div class="pen-title">

</div>
<!-- Form Module-->
```

```
<div class="module form-module">
  <div class="toggle"><i class="fa fa-
  times fa-pencil"></i>
    <div class="tooltip">Click here for
     New User</div>
  </div>
  <div class="form">
    <h2>User Login</h2>
     <form name="f1" id="login" method="post"
     action="<%=request.getContextPath()%>/
     select_identytocken.jsp">
      <input type="text" name="username" id=
      "login_username" placeholder="UserName"/>
      <input type="password" name="password"
      id="login_password" placeholder="Password"/>
      <button>Login</button>
      <br>

                                  <p
             <a href="index.jsp">Click here </a>for
             Admin Login</p>

                                     <br>
                                  <p>
              <a href="index1.jsp">Click here </a>
              forall DataOwner Login</p>
    </form>
  </div>
  <div class="form">
    <h2>Create an account</h2>
        <form action="<%=request.getContextPath()%>/
        NewUser" method="post">
      <input type="text" placeholder="UserID"
      name="userid" />
      <input type="password" placeholder="Pass
      word" name="password"/>
      <input type="text" placeholder="User Name"
       name="name"/>
     <select  name="domain" required="yes" >
                                  <option value="">Domain Code<
                                  /option>
```

```
                                                    <%

        String value;
        String name="";
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;
        UserDAO userDAO=null;
        DAO dao=DAO.getInstance();
        connection=dao.connector();
        statement = connection.createStatement();
        resultSet = statement.executeQuery("select
        domain_code,domain_name from m_domain");
                while(resultSet.next())
                {
                        value=resultSet.getString(1);
                        name=resultSet.getString(2);
                        %>

                        <option value="<%=value%>" ><%=name%>
                        </option>
                        <%

                }


        %>




                                    </select>

        <select  name="subdomain" required="yes">
<option value="">Sub Domain Code</option>
                                                    <%

        String sub_code;
        String sd_name="";
        UserDAO userDAO1=null;
        DAO dao1=DAO.getInstance();
```

```
        connection=dao.connector();
        statement = connection.createStatement();
        resultSet = statement.executeQuery("select
        sdomain_code,sdomain_name from m_subdomain");
                while(resultSet.next())
                {
                        sub_code=resultSet.getString(1);
                        sd_name=resultSet.getString(2);
                        %>

                        <option value="<%=sub_code%>" style=
                        "width: 300px;"><%=sd_name%></option>
                        <%

                }


        %>


                                </select>
                                <input type="email" name="email"
                                required="yes" placeholder="Enter
                                 Your Email" pattern="\w+([-+.]\w+)
                *@\w+([-.]\w+)*\.\w+([-.]\w+)*"></input>
      <button>Register</button>


    </form>
  </div>

</div>
  <script src='http://cdnjs.cloudflare.com/ajax/
  libs/jquery/2.1.3/jquery.min.js'></script>
<script src='http://codepen.io/andytran/pen/vLmRVp.js'>
</script>

      <script src="js/index.js"></script>

</body>
</html>
```

Chapter 7

Results

- This is the admin page where admin logs in and data owner gets registered.
- This is the data owner page where data owner logs in to upload file.
- This is the user login page where user logs in to download the requested file.
- Admin registers the data owner first so that he or she could provide files to users.
- Data owner logs in and uploads requested file into the cloud .
- File uploaded by data owner gets stored in cloud. here we have used DriveHQ as cloud service provider.
- Data owner specifies the access control rights by specifying domain and sub domain of uploaded file.
- Now if a user is valid user i.e matching domain and sub domain of uploaded file, he or she will recieve an identity key at his or her mail.
- If a new user wants to sign up, they will have to get registered first.
- User will log in and upload the identity key mailed to him or her in oreder for verification.

- After this ,user will request for file access from data owner. Data owner will recieve the request.

- Data owner will send the secret key to the user if he or she is valid.

- User will download the secret key and upload it to decrypt the file provided by data owner.

- The decrypted file is downloaded from cloud .

- Downloaded decrypted file is displayed to user.

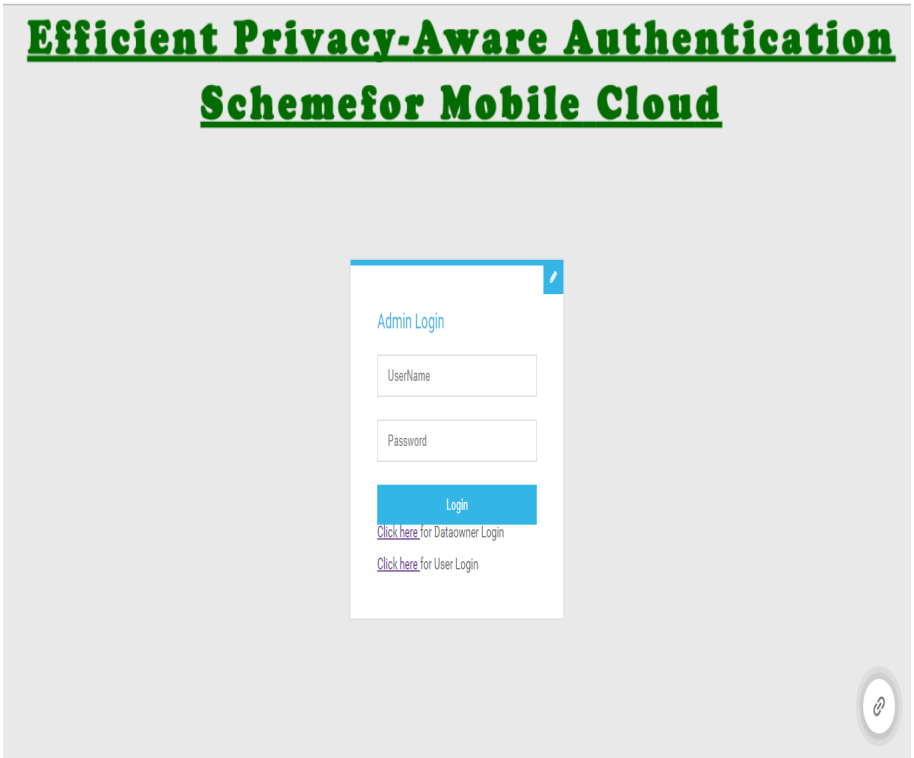

Figure 7.1: Admin page

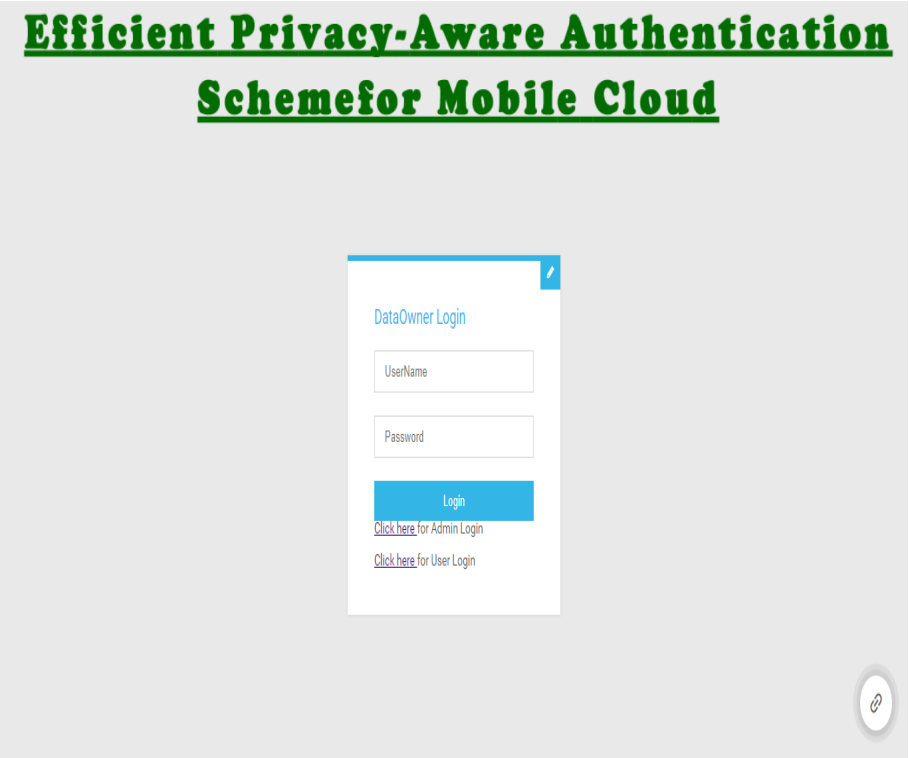

Figure 7.2: Data Owner

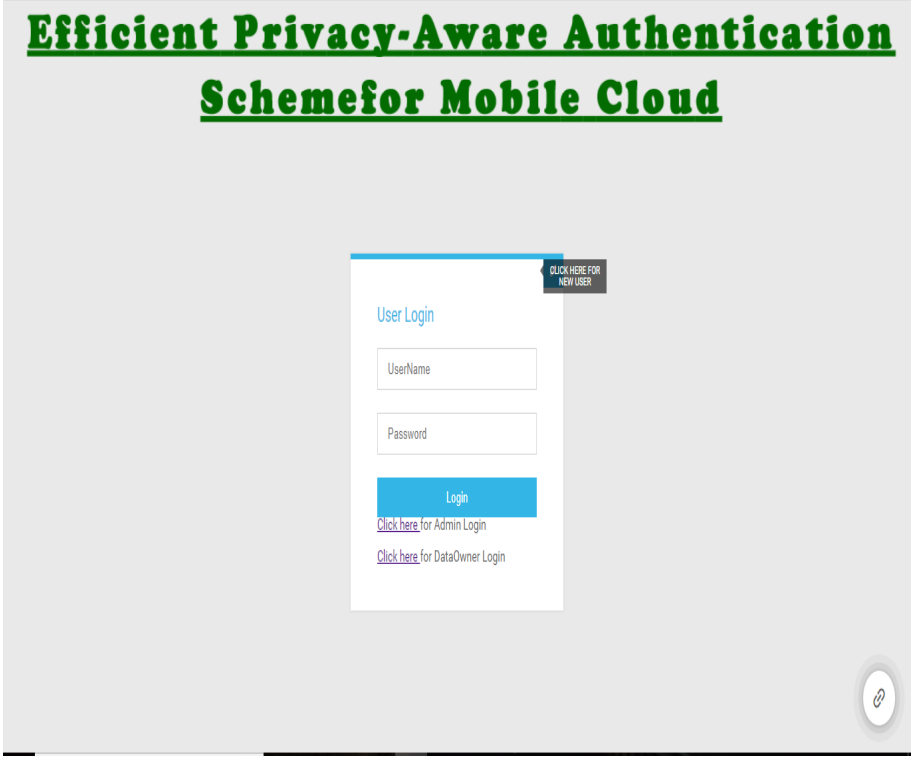

Figure 7.3: User Login

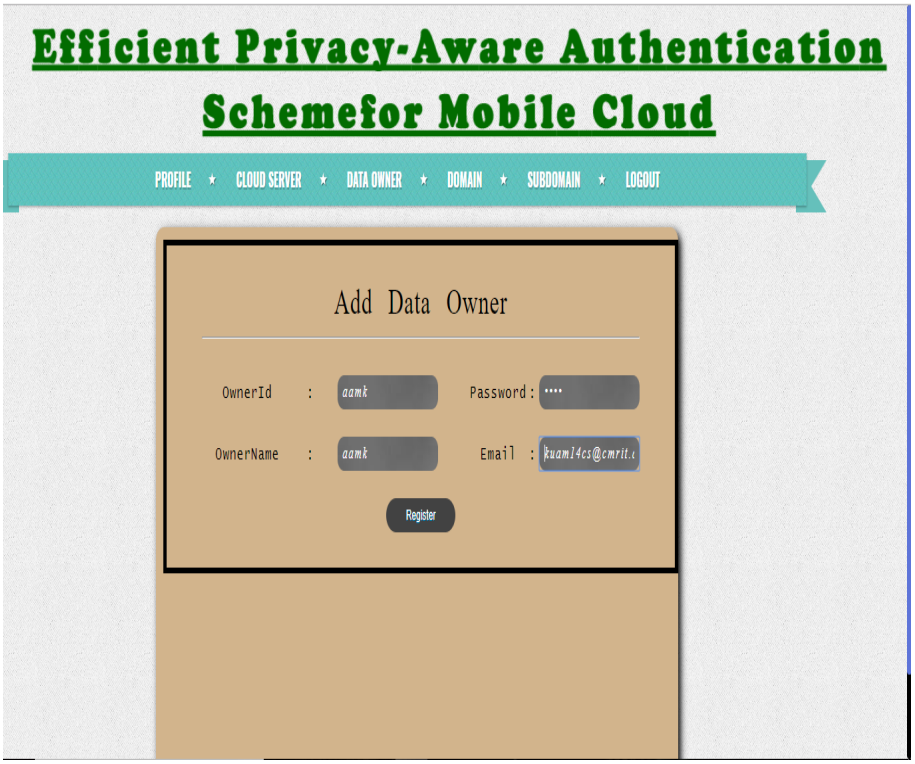

Figure 7.4: Add Data Owner By Admin

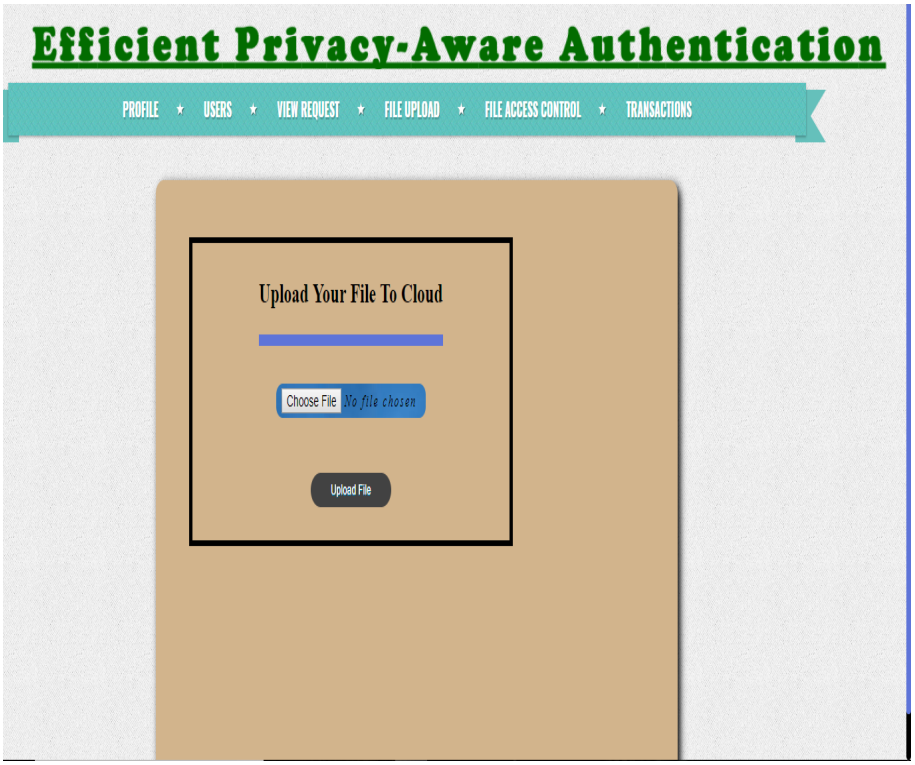

Figure 7.5: File upload by Data Owner

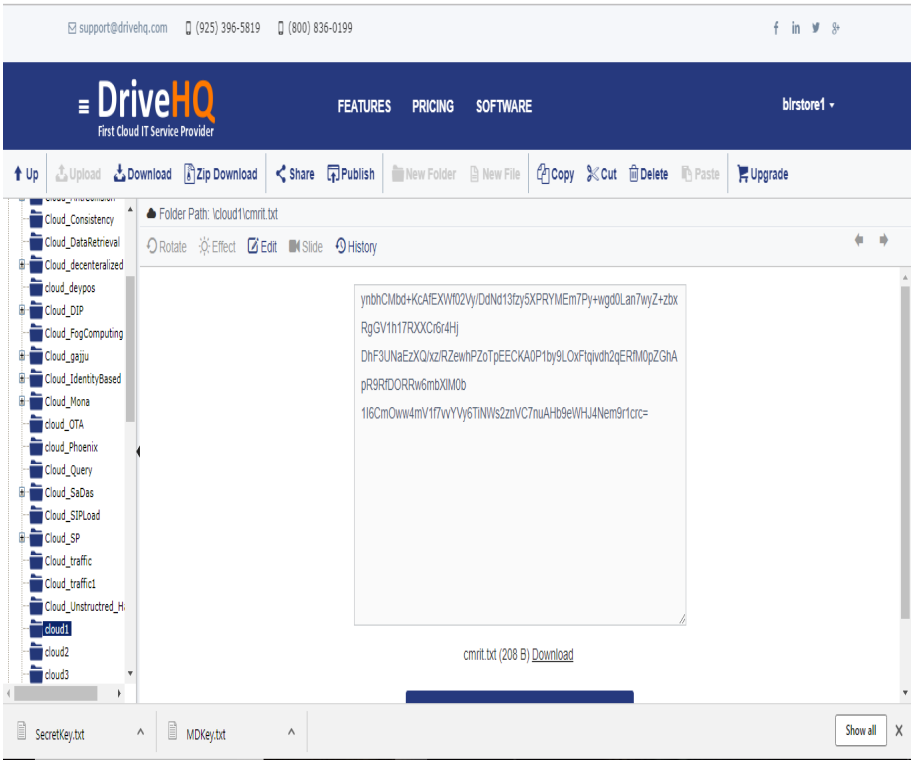

Figure 7.6: Encrypted file in cloud

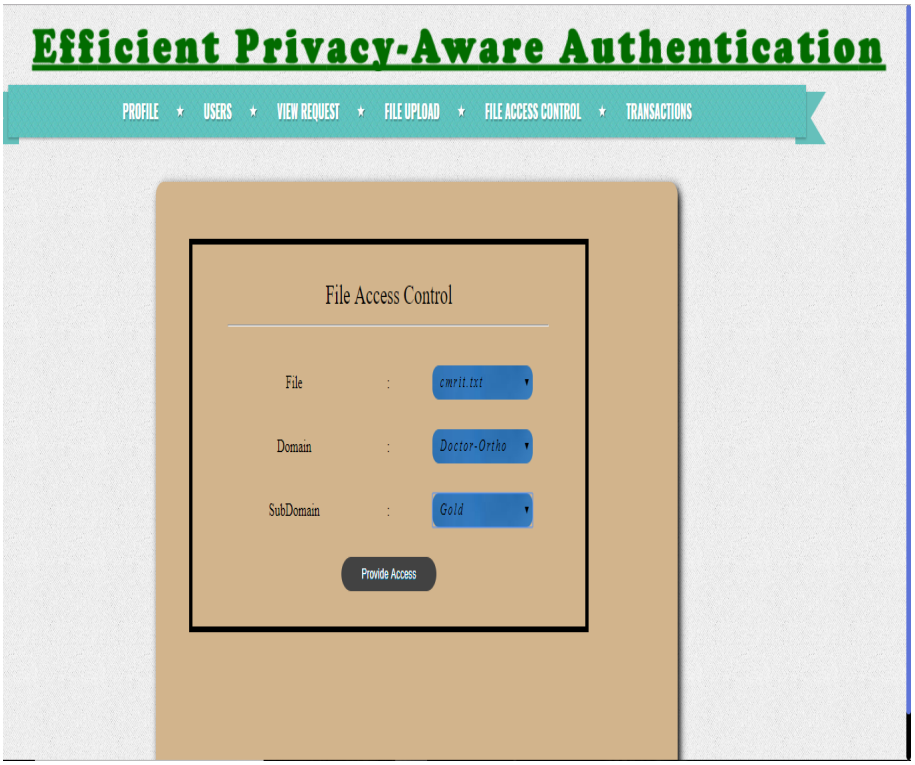

Figure 7.7: File access setup for user

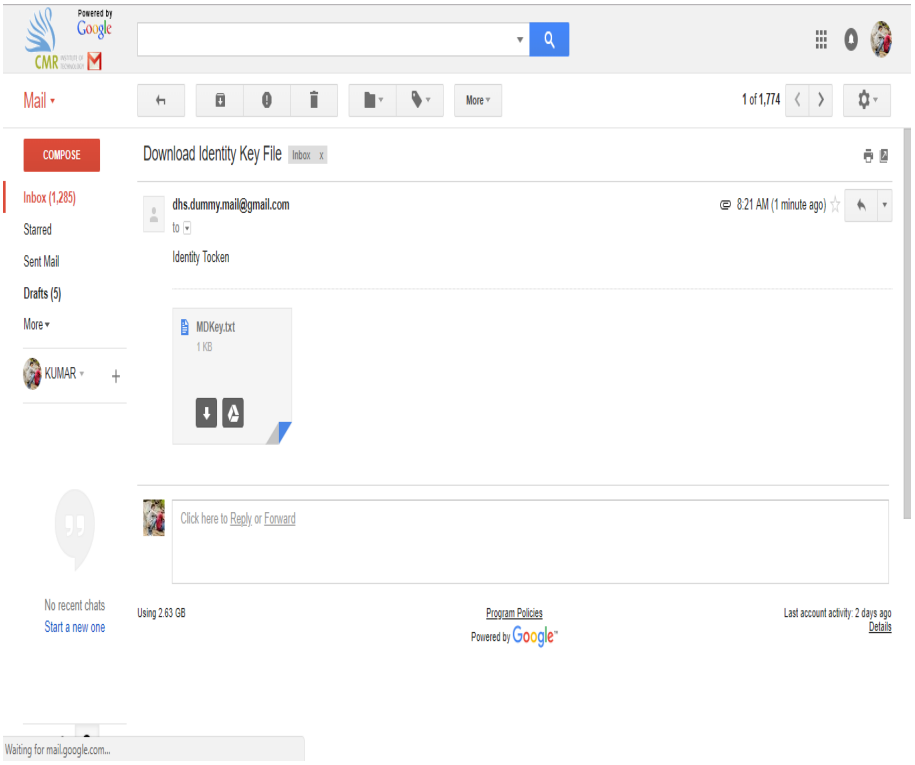

Figure 7.8: Identity token sent to user

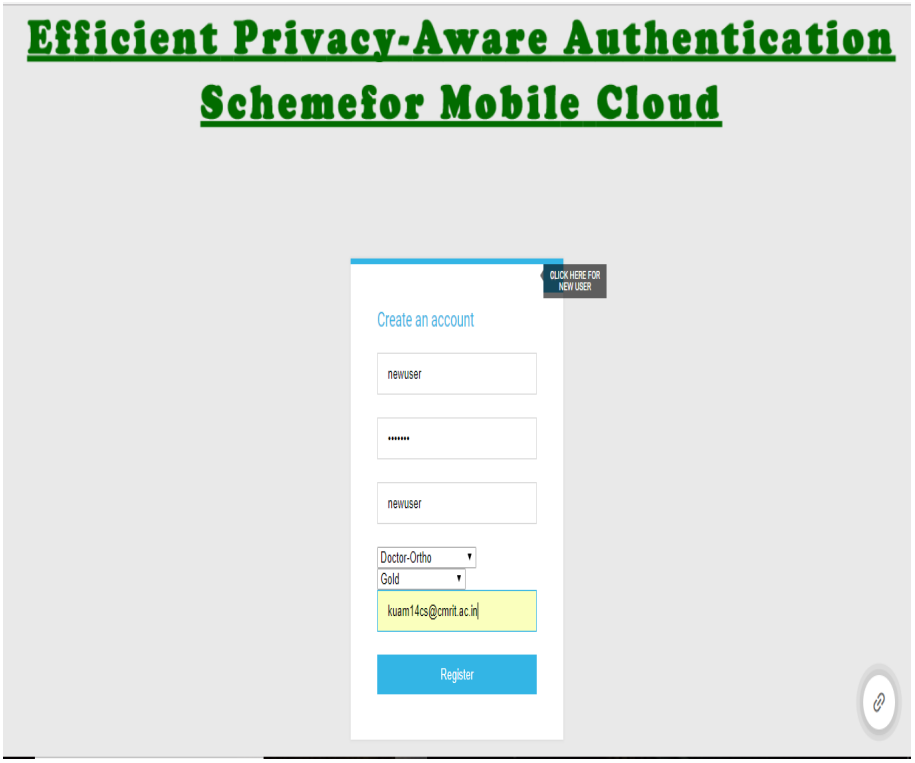

Figure 7.9: New User Signup

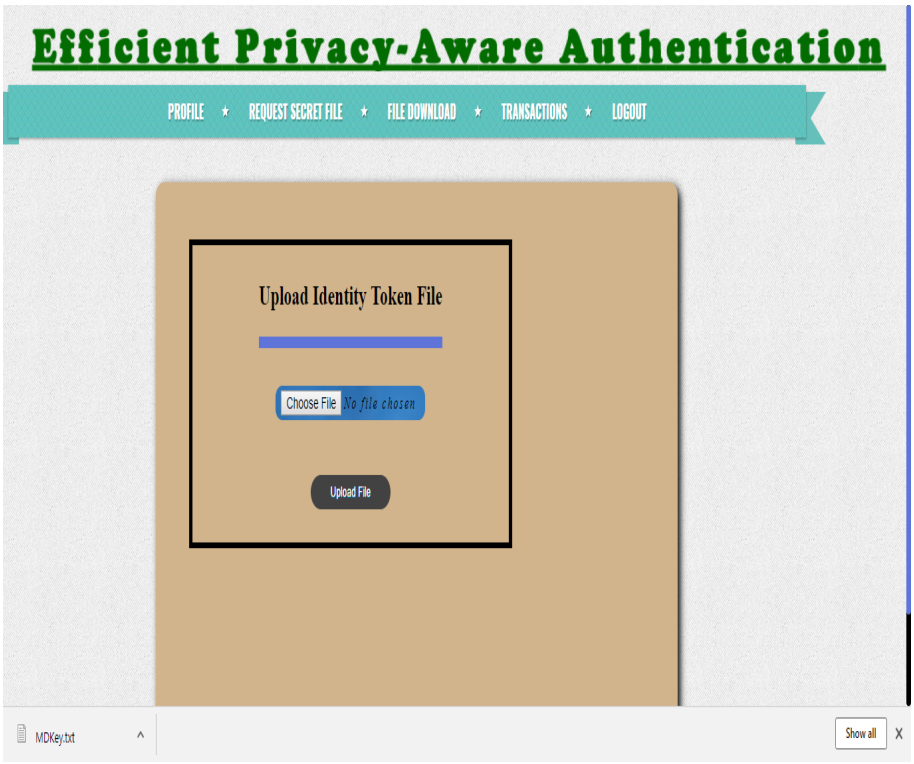

Figure 7.10: File access request by Identity

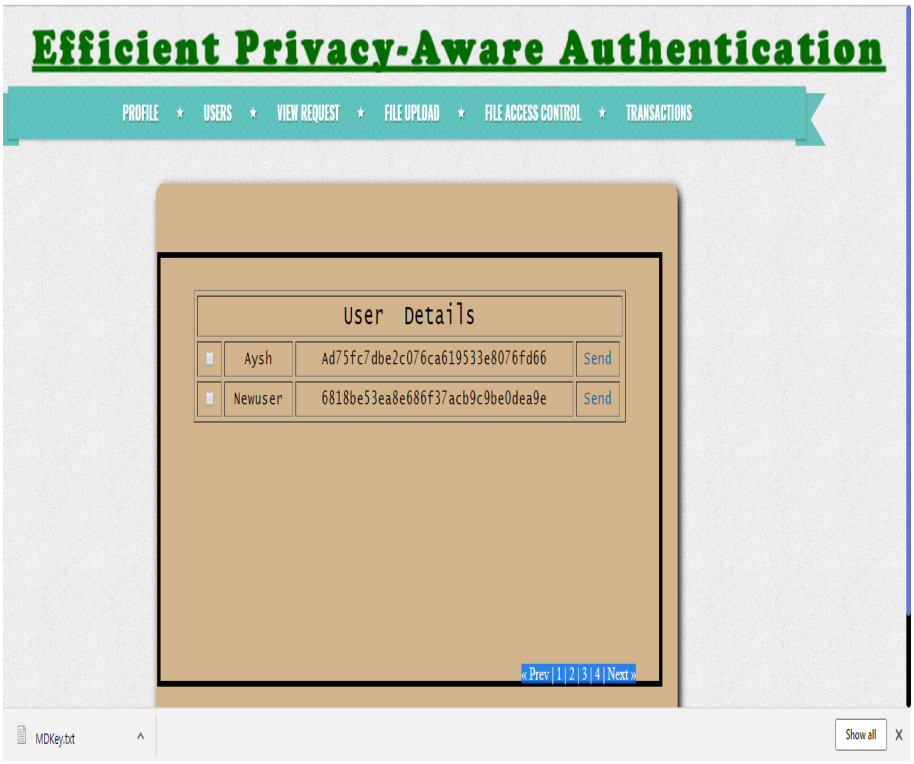

Figure 7.11: Access Request to Data Owner

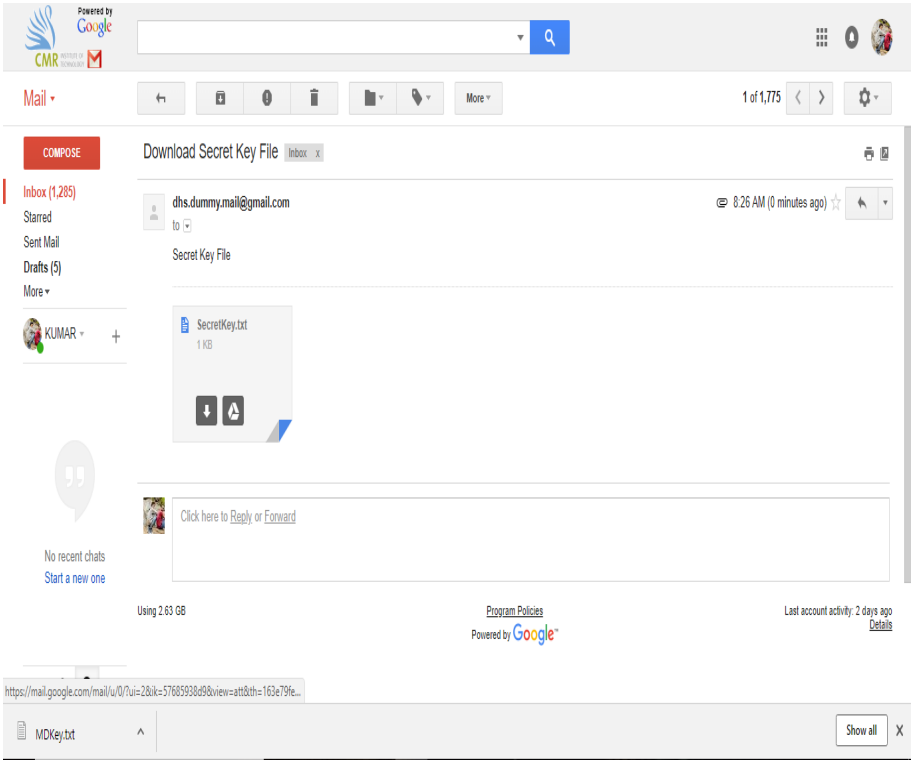

Figure 7.12: Secret Key For File Access

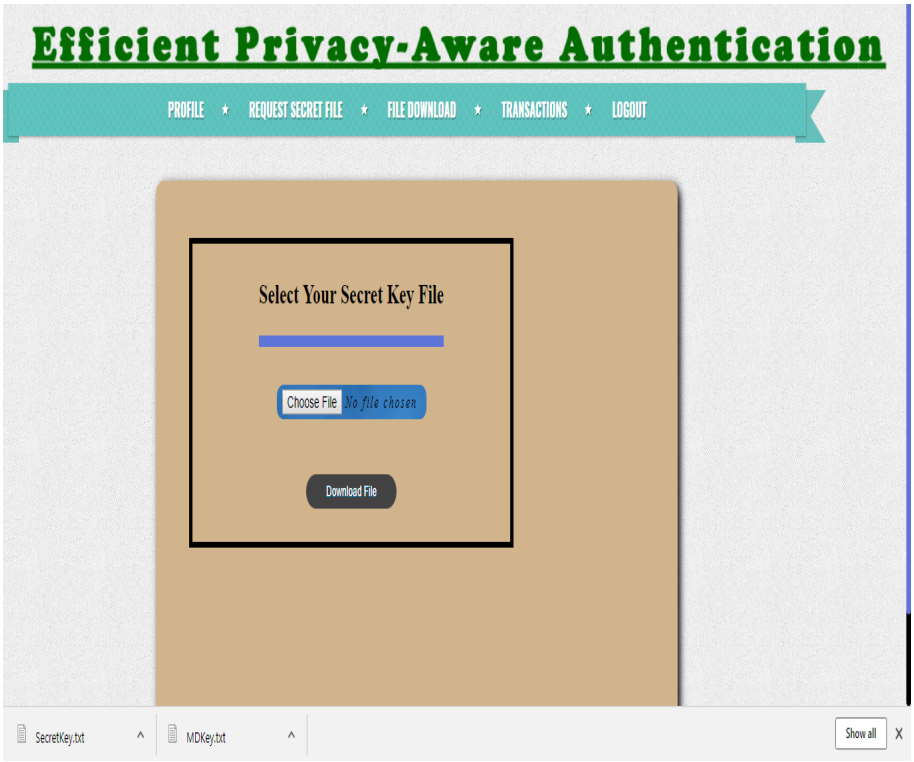

Figure 7.13: Secret key upload To Download File

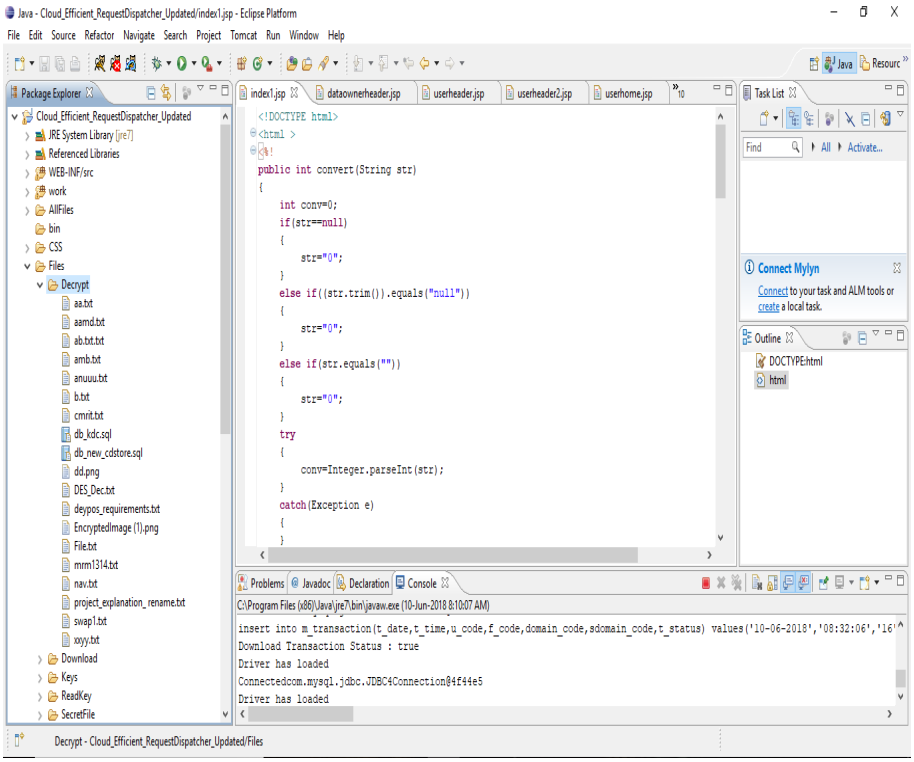

Figure 7.14: Downloaded File

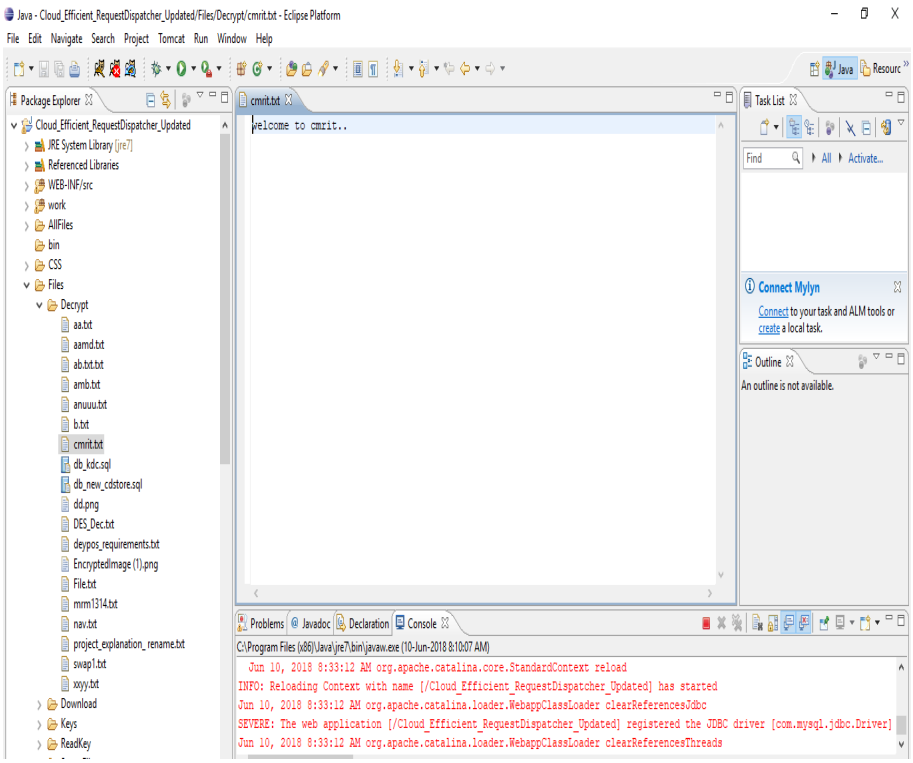

Figure 7.15: Downloaded File From Data Owner

Chapter 8

Conclusion And Future Scope

8.1 Conclusion

Due to highly dynamic nature of mobile devices in the MCC environment, the traditional authentication schemes are not suitable for various services in this environment. To solve the security problem in MCC services, Tsai and Lo proposed an efficient PAA scheme for the MCC services by using the bilinear pairing. This paper points out that Tsai and Los PAA scheme is vulnerable to a serious attack and is not able to support user anonymity. To solve such serious weaknesses, the paper proposes a new PAA scheme for MCC services. Security analysis shows that our proposed PAA scheme can solve the security problem existing in Tsai and Los PAA scheme. Besides, the performance analysis shows that our proposed PAA scheme has better performance than their PAA scheme. In the future, we will explore more attributes of the proposed scheme, which can be applied for secure service access in MCC environment.

8.2 Future Scope

Cloud Computing security challenges are part of ongoing research. Various open issues are identified as future scope.

- Data Classification based on Security:
  A cloud computing data center can store data from various users. To provide the level of security based on the importance of data, classification of data can be done. This classification scheme should consider various aspects like access frequency, update frequency and access by various entities etc. based on the type of data. Once the data is classified and tagged, then level of security associated with this specific tagged data element can be applied. Level of security includes confidentiality, encryption, integrity and storage etc. that are selected

based on the type of data.

- Identity management system:
  Cloud computing users are identified and used their identities for accessing the services. A secure trust based identity management scheme is essentially a need by all cloud service provider and users. Various issues of identity management system are identified. Solution to secure id-generation and distribution, storage and life cycle management is a demand for trust based identity management system.

- Secure trust based Solution for cloud computing Service: A secure environment for execution of the cloud computing services along with overall security considerations is a challenge. A secure and trusted solution is the requirement that needs to be focused and addressed by the cloud computing infrastructure.

- Optimization of resource Utilization:
  Security considerations and provisions for virtualization along with the optimum use of the cloud infrastructure also needs to be focused and addressed.

References

[1] About IPTV on Wikipedia `http://en.wikipedia.org/wiki/DES`

[2] About SMTP on Wikipedia `http://en.wikipedia.org/wiki/SMTP`

[3] About FTP on Wikipedia `http://en.wikipedia.org/wiki/FTP`

[4] About MVC on Wikipedia `http://en.wikipedia.org/wiki/MVC`

[5] [1] M. Satyanarayanan, Fundamental challenges in mobile computing, in Proc. 15th Annu. ACM Symp. Princ. Distrib. Comput., 1996, pp. 17.

[6] [2] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing, IEICE Trans. Commun., vol. 98, no. 1, pp. 190200, 2015.

[7] [3] Z. Xia, X. Wang, X. Sun, and Q. Wang, A secure and dynamic multikeyword ranked search scheme over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 2, pp. 340352, Feb. 2016.

[8] [4] M. Armbrust et al., A view of cloud computing, Commun. ACM, vol. 53, no. 4, pp. 5058, 2010.

[9] [5] A. Lin and N.-C. Chen, Cloud computing as an innovation: Percepetion, attitude, and adoption, Int. J. Inf. Manag., vol. 32, no. 6, pp. 533540, 2012.

[10] [6] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, Enabling personalized search over encrypted outsourced data with efficiency improvement, IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 9, pp. 25462559, Sep. 2016.

[11] [7] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, Mutual verifiable provable data auditing in public cloud storage, J. Internet Technol., vol. 16, no. 2, pp.

317323, 2015.

[12] [8] L. Lamport, Password authentication with insecure communication, Commun. ACM, vol. 24, no. 11, pp. 770772, 1981.

[13] [9] E.-J. Yoon, K.-Y. Yoo, C. Kim, Y.-S. Hong, M. Jo, and H.-H. Chen, A secure and efficient sip authentication scheme for converged VOIP networks, Comput. Commun., vol. 33, no. 14, pp. 16741681, 2010.

[14] [10] R. Arshad and N. Ikram, Elliptic curve cryptography based mutual authentication scheme for session initiation protocol, Multimedia Tools Appl., vol. 66, no. 2, pp. 165178, 2013.

[15] [11] S. H. Islam and G. Biswas, Design of improved password authentication and update scheme based on elliptic curve cryptography, Math. Comput. Modelling, vol. 57, no. 11, pp. 27032717, 2013.

[16] [12] P. Guo, J. Wang, X. Geng, S. K. Chang, and J.-U. Kim, A variable threshold-value authentication architecture for wireless mesh networks, J. Internet Technol., vol. 15, no. 6, pp. 929935, 2014.

[17] [13] J. Shen, H. Tan, J. Wang, J. Wang, and S. Lee, A novel routing protocol providing good transmission reliability in underwater sensor networks, J. Internet Technol., vol. 16, no. 1, pp. 171178, 2015.

[18] [14] M.-S. Hwang and L.-H. Li, A new remote user authentication scheme using smart cards, IEEE Trans. Consum. Electron., vol. 46, no. 1, pp. 28 30, Feb. 2000.

[19] [15] M. S. Farash and M. A. Attari, An anonymous and untraceable password-based authentication scheme for session initiation protocol using smart cards, Int. J. Commun. Syst., vol. 29, no. 13, pp. 19561967, 2016.

[20] [16] A. Irshad, M. Sher, M. S. Faisal, A. Ghani, M. Ul Hassan, and S. Ashraf Ch, A secure authentication scheme for session initiation protocol by using ECC on the basis of the tang and LIU scheme, Secur. Commun. Netw., vol. 7, no. 8, pp. 12101218, 2014.

[21] [17] H.-M. Sun, An efficient remote use authentication scheme using smart cards, IEEE Trans. Consum. Electron., vol. 46, no. 4, pp. 958961, Nov. 2000.

[22] [18] J.-L. Tsai, T.-C. Wu, and K.-Y. Tsai, New dynamic ID authentication scheme using smart cards, Int. J. Commun. Syst., vol. 23, no. 12, pp. 1449 1462, 2010.

[23] [19] C.-T. Li, C.-C. Lee, and C.-W. Lee, An improved two-factor user authentication protocol for wireless sensor networks using elliptic curve cryptography, Sensor Lett., vol. 11, no. 5, pp. 958965, 2013.

[24] [20] S. H. Islam and G. Biswas, Dynamic ID-based remote user mutual authentication scheme with smartcard using elliptic curve cryptography, J. Electron., vol. 31, no. 5, pp. 473488, 2014.

[25] [21] M. S. Farash, S. Kumari, and M. Bakhtiari, Cryptanalysis and improvement of a robust smart card secured authentication scheme on SIP using elliptic curve cryptography, Multimedia Tools Appl., vol. 75, no. 8, pp. 44854504, 2016.

[26] [22] C.-L. Hsu, Y.-H. Chuang, and C.-l. Kuo, A novel remote user authentication scheme from bilinear pairings via internet, Wireless Pers. Commun., vol. 83, no. 1, pp. 163174, 2015.

[27] [23] A. Irshad, M. Sher, E. Rehman, S. A. Ch, M. U. Hassan, and A. Ghani, A single round-trip sip authentication scheme for voice over internet protocol using smart card, Multimedia Tools Appl., vol. 74, no. 11, pp. 39673984, 2015.

[28] [24] A. K. Das, A secure and robust password-based remote user authentication scheme using smart cards for the integrated EPR information system, J. Med.

Syst., vol. 39, no. 3, pp. 114, 2015.

[29] [25] D. Mishra, On the security flaws in id-based password authentication schemes for telecare medical information systems, J. Med. Syst., vol. 39, no. 1, pp. 116, 2015.

[30] [26] L.-H. Li, L.-C. Lin, and M.-S. Hwang, A remote password authentication scheme for multiserver architecture using neural networks, IEEE Trans. Neural Netw., vol. 12, no. 6, pp. 14981504, Nov. 2001.