**A**

# **PROJECT REPORT**

## **ON**

## **A LIGHTWEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING**

**Submitted in partial fulfillment of the requirements For the**

**award of the degree**

## **BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by**

**SK. FAYAZ (18R61A0536)**

**K. JASWANTH (18R61A0518)**

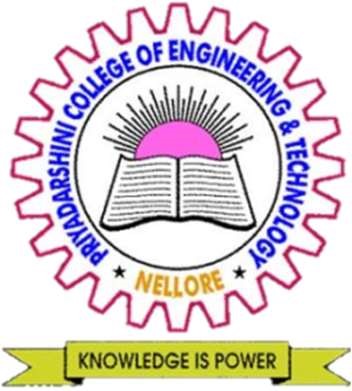
**G. JAGADEESH (18R61A0516)**

**Under the esteemed guidance**

**Mr. A. RAVI M.TECH.,**

**Assistant Professor**

**Department of C.S.E**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**PRIYADARSHINI COLLEGE OF ENGINEERING AND TECHNOLOGY**

**(Affiliated to J.N.T.U.A, Anantapur and approved by A.I.C.T.E., New Delhi)**

## **KANUPARTHIPADU, NELLORE-524004**

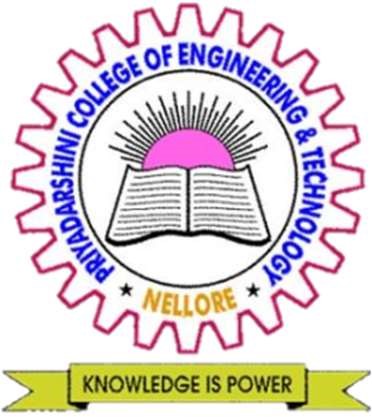
**2018 - 2022**

**PRIYADARSHINI COLLEGE OF ENGINEERING AND**

**TECHNOLOGY**

(Affiliated to J.N.T.U.A, Anantapur and approved by A.I.C.T.E.,New Delhi) KANUPARTHIPADU, NELLORE-524004

COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that project report entitled “A LIGHTWEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING”

Submitted In The partial fulfillment of the requirements for the award of the degree of Bachelor of Technology In Computer Science & Engineering Is a result of the bonafide work carried by the

**SK.FAYAZ (18R61A0536)**

**K.JASWANTH SAI (18R61A0518)**

**G.JAGADEESH (18R61A0516)**

This work has been carried out under my guidance.

Project Guide Head of the Department

Mr.A.RAVI , M.TECH., Dr.D.NAGARAJU,M.TECH,Ph.D

Assistant Professor Head of the department

Dept of CSE Dept of CSE

Submitted for the Viva-voice examination held on

Internal Examiner External Examiner

# **ACKNOWLEDGEMENT**

We express our sincere thanks to Dr. M.MARUTHI PRASADREDDY, M.Tech.,Ph.D., our beloved principal for his valuable suggestions and enormous support during the project.

It is a great pleasure in expressing our deep sense of gratitude and veneration to HOD, Dr.D.NAGARAJU,M.Tech, Ph.D, Head of the Department of Computer science and engineerng for his excellent support and immense help in giving the project shape and success.

We extend our deep sense of gratitude, pleasure and gratefulness to acknowledge Mr.A.RAVI,M.Tech Assistant Professor,Computer Science & Engineering for her excellent guidance help in giving the project shape and success.

We are indebted to the Management for providing us an opportunity with excellent academic, infrastructural facilities to carry out B .Tech program successful.

We are very thankful to teaching and non-teaching staff for their support and guidance throughout our course. At last, but not least we would like to thank our parents, all our friends and well wishers whose constant encouragement and faith in our abilities inspired to deliver our best.

## PROJECT ASSOSICATES

**SK.FAYAZ (18R61A0536)**

**K.JASWANTH SAI (18R61A0518)**

**G.JAGADEESH (18R61A0516)**

**DECLARATION**

I declare that this project report titled “A LIGHTWEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING” submitted in partial fulfilment of the degree of BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING is a record of original work carried out by me under the supervision of Mr.A.RAVI,.M..Tech and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

DATE:

PLACE:

## PROJECT ASSOSICATES

**SK.FAYAZ (18R61A0536)**

**K.JASWANTH SAI (18R61A0518)**

**G.JAGADEESH (18R61A0516)**

## **ABSTRACT**

TITLE: A LIGHT WEIGHT SECURE DATA SHARING SCHEME FOR MOBILE CLOUD COMPUTING

With the popularity of cloud computing, mobile devices can store/retrieve personal data from anywhere at any time.

Consequently, the data security problem in mobile cloud becomes more and more severe and prevents further development of mobile cloud.

There are substantial studies that have been conducted to improve the cloud security. However, most of them are not applicable for mobile cloud since mobile devices only have limited computing resources and power.

Solutions with low computational overhead are in great need for mobile cloud applications.

In this paper, we propose a lightweight data sharing scheme (LDSS) for mobile cloud computing.

It adopts CP-ABE, an access control technology used in normal cloud environment, but changes the structure of access control tree to make it suitable for mobile cloud environments. LDSS moves a large portion of the computational intensive access control tree transformation in CP-ABE from mobile devices to external proxy servers.

Furthermore, to reduce the user revocation cost, it introduces attribute description fields to implement lazy-revocation, which is a thorny issue in program based CP-ABE systems. The experimental results show that LDSS can effectively reduce the overhead on the mobile device side when users are sharing data in mobile cloud environments.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Chapter No | Chapter Name With Topic wise | |  | Page NO |
| 1 | Introduction | |  | 8-13 |
| 2 | Literature Survey | |  | 14-18 |
| 3 | Systems Analysis | |  | 19-21 |
| 3.1 | Existing System |  | 20 |
| 3.1.1 | Disadvantages of existing system |  | 20 |
| 3.2 | Proposed System |  | 21 |
| 3.2.1 | Advantages of proposed system |  | 21 |
| 3.3 | System Requirements |  | 22 |
| 3.3.1 | Hardware requirements |  | 22 |
| 3.4 | Attribute-based fine-grained access control with efficient revocation in cloud storage systems |  | 15 |
| 3.4.1 | Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data |  | 16 |
| 3.4.2 | DACMACS: Effective Data Access Control for Multiauthority Cloud Storage Systems |  | 17 |
| 3.4.3 | Attribute based proxy re-encryption with delegating capabilities |  | 18 |
| 3.4.4 | Attribute based proxy re-encryption with delegating capabilities AUTHORS |  | 18 |
| 4 | System Design | |  | 23-32 |
| 4.2 | Data flow Diagram |  | 24 |
| 4.4 | Class Diagrams |  | 26 |
| 4.5 | Use case Diagrams |  | 27 |
| 4.6 | Sequence Diagram |  | 28 |
| 4.7 | Deployement Diagram |  | 29 |
| 4.4 | Collaboration Diagram |  | 30 |
|  | 4.5 | Component Diagram |  | 30 |
|  | 4.5 | State Diagram |  | 31 |
| 5 | System Implementation | |  | 33-41 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 | Software environment | | 42-62 | |  |
|  | 6.1.2 | Java technologies | | 42-52 | | |  |
|  | 6.1.3 | Networking | | 53-62 | | |  |
| 7 | System testing | | 63-67 | |  |
| 7.1 | System Testing | | 64 | | |  |
| 7.2 | Types of testing | | 64-67 | | |
| 8 |  | SYSTEM STUDY | | 68-69 | | |  |
| 9 |  | Output screenshots | | 70-75 | | |  |
| 10 |  | Conclusion | | 76 | | |
| 11 |  | Bibliography | | 77-81 | | |

# INTRODUCTION

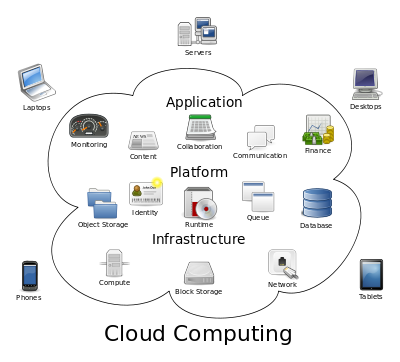
**CHAPTER 1**

**INTRODUCTION**

**1.Introduction**

**What is cloud computing?**

**Cloud computing** is the use of [computing](http://en.wikipedia.org/wiki/Computing) resources (hardware and software) that are delivered as a service over a network (typically the [Internet](http://en.wikipedia.org/wiki/Internet)). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.



Structure of cloud computing

**How Cloud Computing Works?**

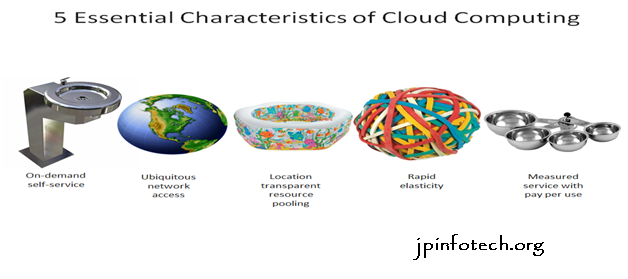
The goal of cloud computing is to apply traditional [supercomputing](http://www.webopedia.com/TERM/S/supercomputer.html), or [high-performance computing](http://www.webopedia.com/TERM/H/High_Performance_Computing.html) power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games.

The cloud computing uses [networks](http://www.webopedia.com/TERM/N/network.html) of large groups of [servers](http://www.webopedia.com/TERM/S/server.html) typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared [IT](http://www.webopedia.com/TERM/I/IT.html) infrastructure contains large pools of systems that are linked together. Often, [virtualization](http://www.webopedia.com/TERM/V/virtualization.html) techniques are used to maximize the power of cloud computing.

**Characteristics and Services Models:**

The salient characteristics of cloud computing based on the definitions provided by the National Institute of Standards and Terminology (NIST) are outlined below:

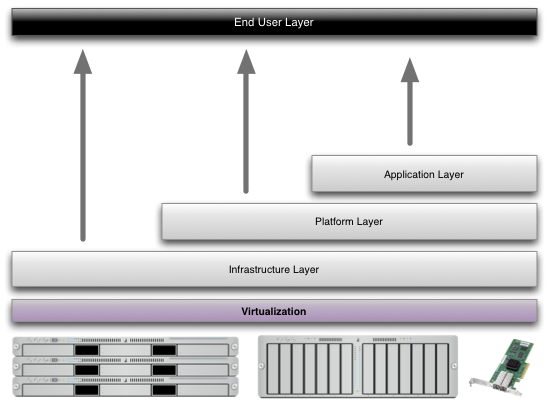
* **On-demand self-service**: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service’s provider.
* **Broad network access**: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
* **Resource pooling**: The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
* **Rapid elasticity**: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
* **Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.



Characteristics of cloud computing

**Services Models:**

  Cloud Computing comprises three different service models, namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The three service models or layer are completed by an end user layer that encapsulates the end user perspective on cloud services. The model is shown in figure below. If a cloud user accesses services on the infrastructure layer, for instance, she can run her own applications on the resources of a cloud infrastructure and remain responsible for the support, maintenance, and security of these applications herself. If she accesses a service on the application layer, these tasks are normally taken care of by the cloud service provider.



Structure of service models

**Benefits of cloud computing:**

1. **Achieve economies of scale** – increase volume output or productivity with fewer people. Your cost per unit, project or product plummets.
2. **Reduce spending on technology infrastructure.** Maintain easy access to your information with minimal upfront spending. Pay as you go (weekly, quarterly or yearly), based on demand.
3. **Globalize your workforce on the cheap.** People worldwide can access the cloud, provided they have an Internet connection.
4. **Streamline processes.** Get more work done in less time with less people.
5. **Reduce capital costs.** There’s no need to spend big money on hardware, software or licensing fees.
6. **Improve accessibility.** You have access anytime, anywhere, making your life so much easier!
7. **Monitor projects more effectively.** Stay within budget and ahead of completion cycle times.
8. **Less personnel training is needed.** It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.
9. **Minimize licensing new software.** Stretch and grow without the need to buy expensive software licenses or programs.
10. **Improve flexibility.** You can change direction without serious “people” or “financial” issues at stake.

**Advantages:**

1. **Price:** Pay for only the resources used.
2. **Security**: Cloud instances are isolated in the network from other instances for improved security.
3. **Performance:** Instances can be added instantly for improved performance. Clients have access to the total resources of the Cloud’s core hardware.
4. **Scalability:** Auto-deploy cloud instances when needed.
5. **Uptime:** Uses multiple servers for maximum redundancies. In case of server failure, instances can be automatically created on another server.
6. **Control:** Able to login from any location. Server snapshot and a software library lets you deploy custom instances.
7. **Traffic:** Deals with spike in traffic with quick deployment of additional instances to handle the load.

# LITERATURUTE SURVEY

**CHAPTER 2**

**LITERATURE SURVEY**

## **LITERATURE SURVEY**

1. **Attribute-based fine-grained access control with efficient revocation in cloud storage systems**

**AUTHORS:**  Kan Yang, Xiaohua Jia, Kui Ren

A cloud storage service allows data owner to outsource their data to the cloud and through which provide the data access to the users. Because the cloud server and the data owner are not in the same trust domain, the semi-trusted cloud server cannot be relied to enforce the access policy. To address this challenge, traditional methods usually require the data owner to encrypt the data and deliver decryption keys to authorized users. These methods, however, normally involve complicated key management and high overhead on data owner. In this paper, we design an access control framework for cloud storage systems that achieves fine-grained access control based on an adapted Ciphertext-Policy Attribute-based Encryption (CP-ABE) approach. In the proposed scheme, an efficient attribute revocation method is proposed to cope with the dynamic changes of users' access privileges in large-scale systems. The analysis shows that the proposed access control scheme is provably secure in the random oracle model and efficient to be applied into practice.

**2) Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data**

**AUTHORS:**  Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje

As the data produced by individuals and enterprises that need to be stored and utilized are rapidly increasing, data owners are motivated to outsource their local complex data management systems into the cloud for its great flexibility and economic savings. However, as sensitive cloud data may have to be encrypted before outsourcing, which obsoletes the traditional data utilization service based on plaintext keyword search, how to enable privacy-assured utilization mechanisms for outsourced cloud data is thus of paramount importance. Considering the large number of on-demand data users and huge amount of outsourced data files in cloud, the problem is particularly challenging, as it is extremely difficult to meet also the practical requirements of performance, system usability, and high-level user searching experiences. In this paper, we investigate the problem of secure and efficient similarity search over outsourced cloud data. Similarity search is a fundamental and powerful tool widely used in plaintext information retrieval, but has not been quite explored in the encrypted data domain. Our mechanism design first exploits a suppressing technique to build storage-efficient similarity keyword set from a given document collection, with edit distance as the similarity metric. Based on that, we then build a private trie-traverse searching index, and show it correctly achieves the defined similarity search functionality with constant search time complexity. We formally prove the privacy-preserving guarantee of the proposed mechanism under rigorous security treatment. To demonstrate the generality of our mechanism and further enrich the application spectrum, we also show our new construction naturally supports fuzzy search, a previously studied notion aiming only to tolerate typos and representation inconsistencies in the user searching input. The extensive experiments on Amazon cloud platform with real data set further demonstrate the validity and practicality of the proposed mechanism.

1. **DACMACS: Effective Data Access Control for Multiauthority Cloud Storage Systems**

**AUTHORS:**  Kan Yang, Xiaohua Jia, Kui Ren, Bo Zhang, Ruitao Xie

Data access control is an effective way to ensure data security in the cloud. However, due to data outsourcing and untrusted cloud servers, the data access control becomes a challenging issue in cloud storage systems. Existing access control schemes are no longer applicable to cloud storage systems, because they either produce multiple encrypted copies of the same data or require a fully trusted cloud server. Ciphertext-policy attribute-based encryption (CP-ABE) is a promising technique for access control of encrypted data. However, due to the inefficiency of decryption and revocation, existing CP-ABE schemes cannot be directly applied to construct a data access control scheme for multiauthority cloud storage systems, where users may hold attributes from multiple authorities. In this paper, we propose data access control for multiauthority cloud storage (DAC-MACS), an effective and secure data access control scheme with efficient decryption and revocation. Specifically, we construct a new multiauthority CP-ABE scheme with efficient decryption, and also design an efficient attribute revocation method that can achieve both forward security and backward security. We further propose an extensive data access control scheme (EDAC-MACS), which is secure under weaker security assumptions.

# 4) Attribute based proxy re-encryption with delegating capabilities.

# AUTHORS: Liang Xiaohui, Cao Zhenfu, Lin Huang

Attribute based proxy re-encryption scheme (ABPRE) is a new cryptographic primitive which extends the traditional proxy re-encryption (public key or identity based cryptosystem) to the attribute based counterpart, and thus empower users with delegating capability in the access control environment. Users, identified by attributes, could freely designate a proxy who can re-encrypt a ciphertext related with a certain access policy to another one with a different access policy. The proposed scheme is proved selective-structure chosen plaintext secure and master key secure without random oracles. Besides, we develop another kind of key delegating capability in our scheme and also discuss some related issues including a stronger security model and applications.

**5)** **Attribute based proxy re-encryption with delegating capabilities**

**AUTHORS:**  Liang Xiaohui, Cao Zhenfu, Lin Huang

Attribute based proxy re-encryption scheme (ABPRE) is a new cryptographic primitive which extends the traditional proxy re-encryption (public key or identity based cryptosystem) to the attribute based counterpart, and thus empower users with delegating capability in the access control environment. Users, identified by attributes, could freely designate a proxy who can re-encrypt a ciphertext related with a certain access policy to another one with a different access policy. The proposed scheme is proved selective-structure chosen plaintext secure and master key secure without random oracles. Besides, we develop another kind of key delegating capability in our scheme and also discuss some related issues including a stronger security model and applications.

# SYSTEM ANALYSIS

**CHAPTER 3**

**SYSTEM ANALYSIS**

## **SYSTEM ANALYSIS**

**EXISTING SYSTEM:**

* In general, we can divide these approaches into four categories: simple ciphertext access control, hierarchical access control, access control based on fully homomorphic encryption and access control based on attribute-based encryption (ABE). All these proposals are designed for non-mobile cloud environment
* Tysowski et al. considered a specific cloud computing environment where data are accessed by resource-constrained mobile devices, and proposed novel modifications to ABE, which assigned the higher computational overhead of cryptographic operations to the cloud provider and lowered the total communication cost for the mobile user.

**DISADVANTAGES OF EXISTING SYSTEM:**

* Data privacy of the personal sensitive data is a big concern for many data owners.
* The state-of-the-art privilege management/access control mechanisms provided by the CSP are either not sufficient or not very convenient.
* They cannot meet all the requirements of data owners.
* They consume large amount of storage and computation resources, which are not available for mobile devices
* Current solutions don’t solve the user privilege change problem very well. Such an operation could result in very high revocation cost. This is not applicable for mobile devices as well. Clearly, there is no proper solution which can effectively solve the secure data sharing problem in mobile cloud.

**PROPOSED SYSTEM:**

* We propose a Lightweight Data Sharing Scheme (LDSS) for mobile cloud computing environment.
* The main contributions of LDSS are as follows:
* We design an algorithm called LDSS-CP-ABE based on Attribute-Based Encryption (ABE) method to offer efficient access control over ciphertext.
* We use proxy servers for encryption and decryption operations. In our approach, computational intensive operations in ABE are conducted on proxy servers, which greatly reduce the computational overhead on client side mobile devices. Meanwhile, in LDSS-CP-ABE, in order to maintain data privacy, a version attribute is also added to the access structure. The decryption key format is modified so that it can be sent to the proxy servers in a secure way.
* We introduce lazy re-encryption and description field of attributes to reduce the revocation overhead when dealing with the user revocation problem.
* Finally, we implement a data sharing prototype framework based on LDSS.

**ADVANTAGES OF PROPOSED SYSTEM:**

* The experiments show that LDSS can greatly reduce the overhead on the client side, which only introduces a minimal additional cost on the server side.
* Such an approach is beneficial to implement a realistic data sharing security scheme on mobile devices.
* The results also show that LDSS has better performance compared to the existing ABE based access control schemes over ciphertext.
* Multiple revocation operations are merged into one, reducing the overall overhead
* In LDSS, the storage overhead needed for access control is very small compared to data files.

**SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

* Operating system : - Windows XP/7.
* Coding Language : JAVA/J2EE
* Data Base : MYSQL

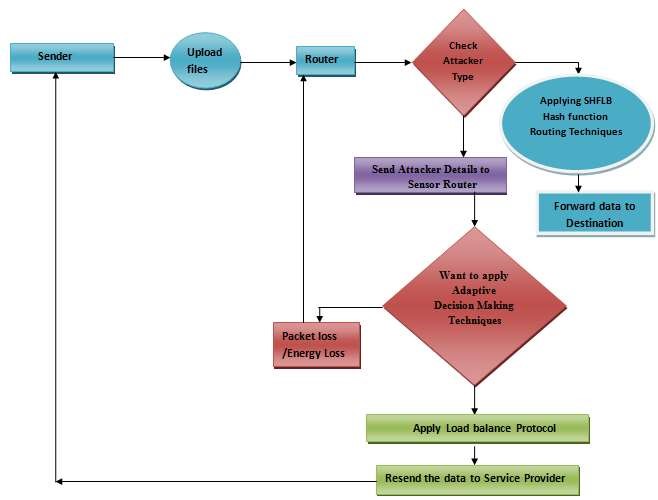
# SYSTEM DESIGN

**CHAPTER 4**

**SYSTEM DESIGN**

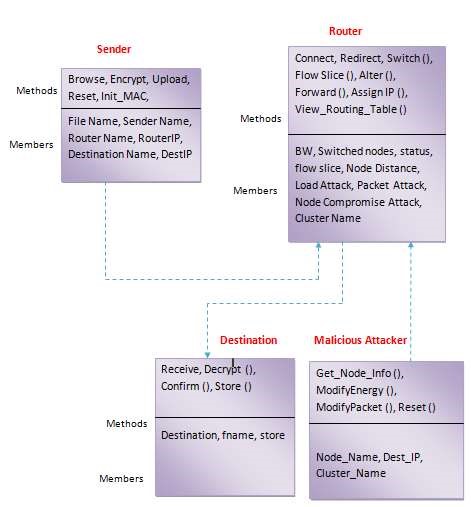
**DATA FLOW DIAGRAM:**

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



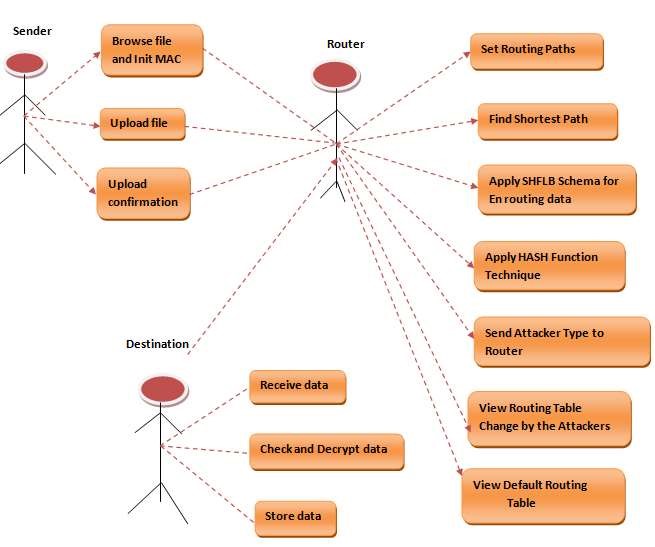
**CLASS DIAGRAM**:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



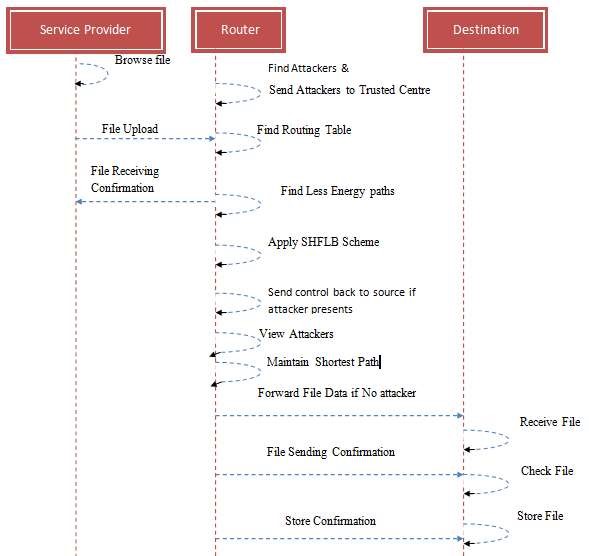
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



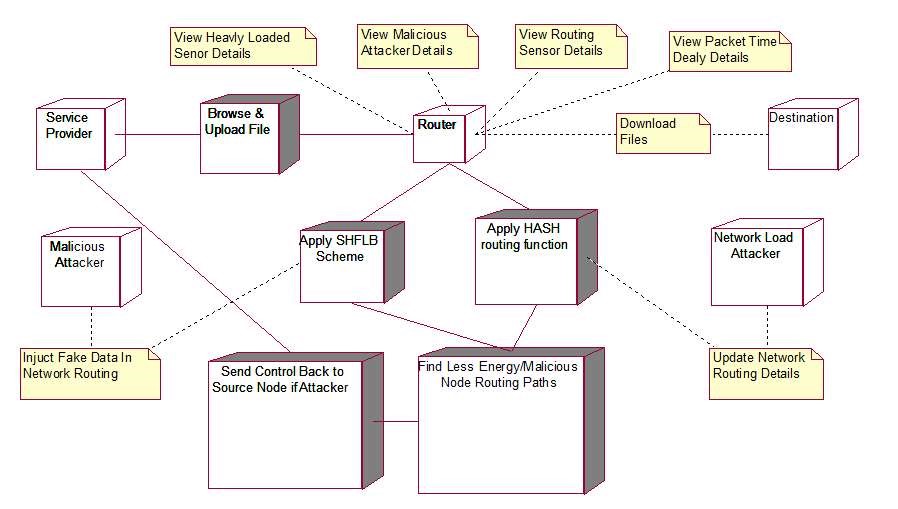
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



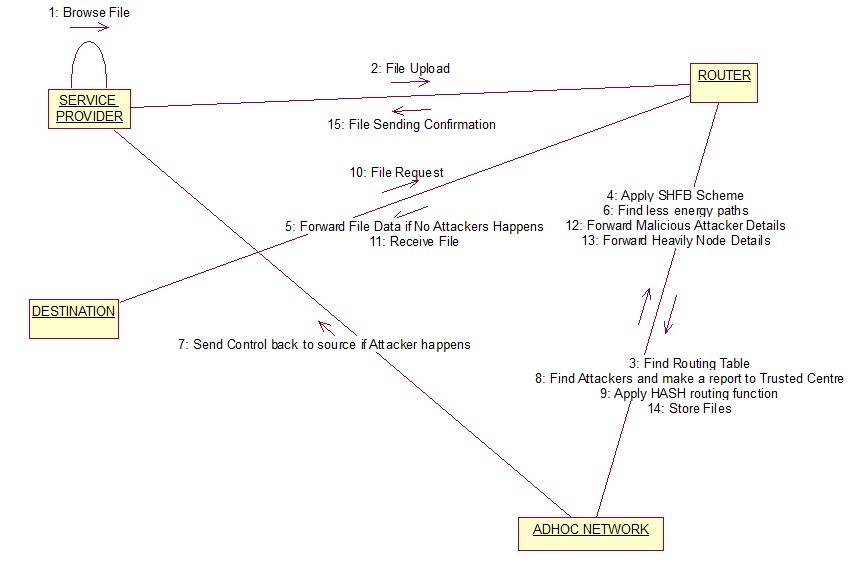
## DEPLOYMENET DIAGRAM

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. A deployment diagram is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware). Deployment diagrams are important for visualizing, specifying, and documenting embedded, client/server, and distributed systems and also for managing executable systems through forward and reverse engineering.



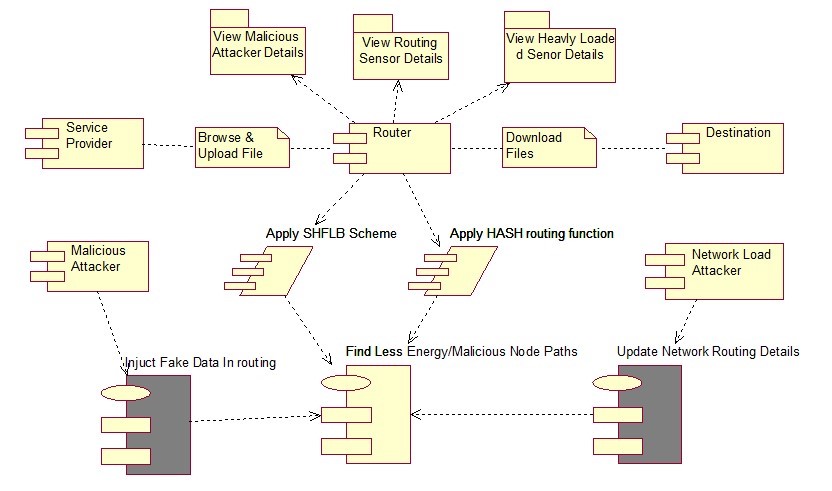
## COLLOBORATION DIAGRAM

Collaboration diagrams (known as Communication Diagram in UML 2.x) are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. Along with sequence diagrams, collaboration are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.



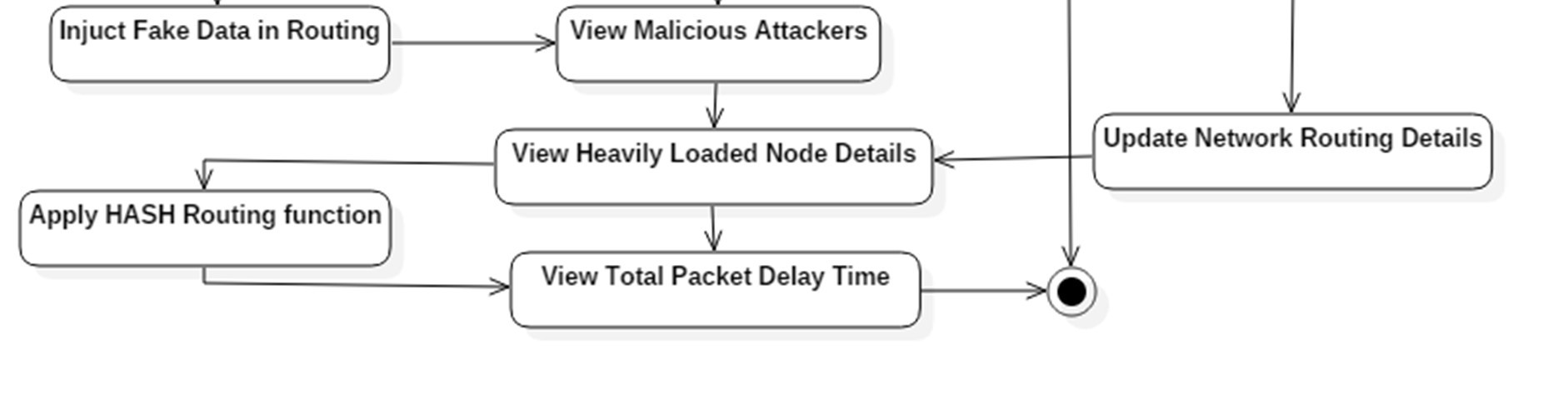
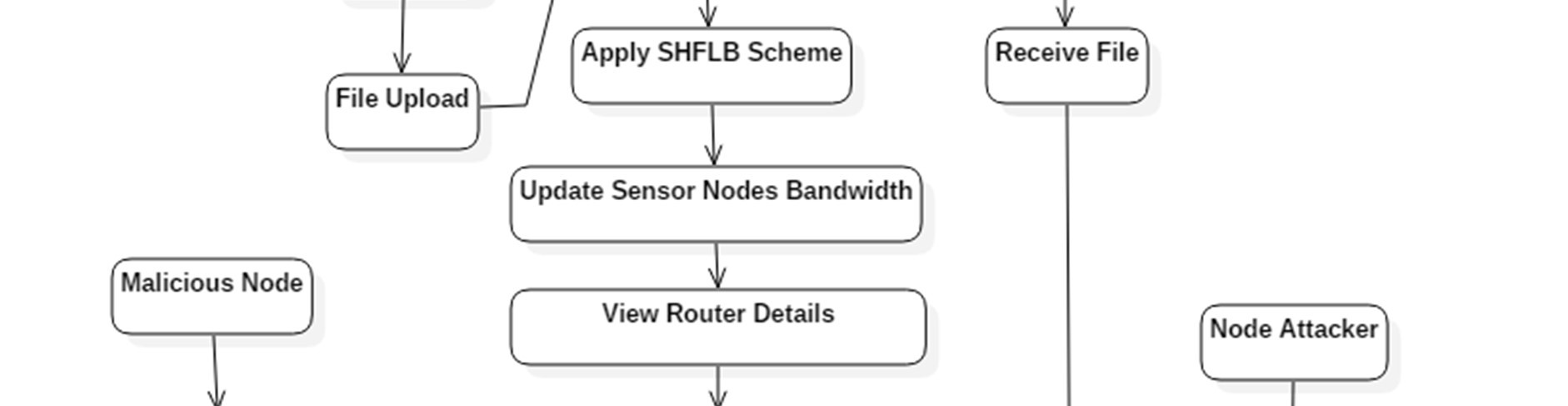
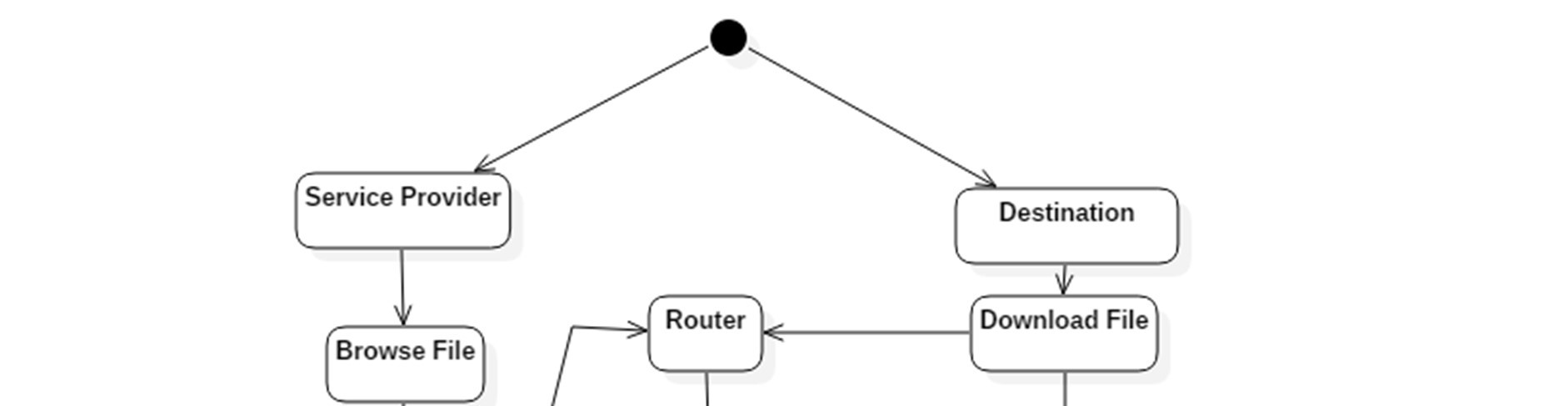
## COMPONENT DIAGRAM

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system. A component diagram breaks down the actual system under development into various high levels of functionality. Each component is responsible for one clear aim within the entire system and only interacts with other essential elements on a need-to-know basis.



## STATE CHART DIAGRAM

A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It’s a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and Statechart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don’t model every class using State diagrams. We prefer to model the states with three or more states.



# SYSTEM IMPLEMENTATION

**CHAPTER 5**

## **SYSTEM IMPLEMENTATION**

### SYSTEM IMPLEMENTATION

**MODULES:**

* System Framework
* Data Owner
* Data User
* Trusted Authority
* Cloud Service Provider

**MODULES DESCRIPTION:**

**System Framework:**

The development of cloud computing and the popularity of smart mobile devices, people are gradually getting accustomed to a new era of data sharing model in which the data is stored on the cloud and the mobile devices are used to store/retrieve the data from the cloud. In these applications, people (data owners) can upload their documents and other files to the cloud and share these data with other people (data users) they like to share. CSPs also provide data management functionality for data owners. Since personal data files are sensitive, data owners are allowed to choose whether to make their data files public or can only be shared with specific data users. Clearly, data privacy of the personal sensitive data is a big concern for many data owners. We propose LDSS, a framework of lightweight data sharing scheme in mobile cloud. It has the following six components. (1)Data Owner (DO) (2) Data User (DU) (3) Trust Authority (TA) (4) Encryption Service Provider (ESP) (5) Decryption Service Provider (DSP) (6) Cloud Service Provider (CSP).

**Data Owner (DO):**

When the data owner (DO) registers on TA, TA runs the algorithm Setup() to generate a public key PK and a master key MK. PK is sent to DO while MK is kept on TA itself. DO defines its own attribute set and assigns attributes to its contacts. All these information will be sent to TA and the cloud. TA and the cloud receive the information and store it. DO uploads data to the mobile cloud and share it with friends. DO determines the access control policies. DO sends data to the cloud. Since the cloud is not credible, data has to be encrypted before it is uploaded. The DO defines access control policy in the form of access control tree on data files to assign which attributes a DU should obtain if he wants to access a certain data file.

**Data User (DU):**

DU logins onto the system and sends, an authorization request to TA. The authorization request includes attribute keys (SK) which DU already has. TA accepts the authorization request and checks the request and a generate attribute keys (SK) for DU. DU sends a request for data to the cloud.Cloud receives the request and checks if the DU meets the access requirement. DU receives the ciphertext, which includes ciphertext of data files and ciphertext of the symmetric key. DU decrypt the ciphertext of the symmetric key with the assistance of DSP. DU uses the symmetric key to decrypt the ciphertext of data files.

**Trusted Authority:**

To make LDSS feasible in practice, a trusted authority (TA) is introduced. It is responsible of generating public and private keys, and distributing attribute keys to users. With this mechanism, users can share and access data without being aware of the encryption and decryption operations. We assume TA is entirely credible, and a trusted channel exists between the TA and every user. The fact that a trusted channel exists doesn’t mean that the data can be shared through the trusted channel, for the data can be in a large amount. TA is only used to transfer keys (in a small amount) securely between users. In addition, it’s requested that TA is online all the time because data users may access data at any time and need TA to update attribute keys.

**Cloud Service Provider:**

CSP stores the data for DO. It faithfully executes the operations requested by DO, while it may peek over data that DO has stored in the cloud. DU sends a request for data to the cloud. Cloud receives the request and checks if the DU meets the access requirement. If DU can’t meet the requirement, it refuses the request; otherwise it sends the ciphertext to DU. CSP manages the Uploaded Files.

import java.io.File;

import java.io.FileInputStream;

import org.apache.commons.net.ftp.FTPClient;

public class Ftpcon {

FTPClient client = new FTPClient();

FileInputStream fis = null;

boolean status;

public boolean upload(File file){

try{

// client.enterLocalPassiveMode();

client.connect("ftp.drivehq.com");

client.login("jaswanthsai3459@gmail.com", "jaswanth-123");

client.enterLocalPassiveMode();

//String filename = "/home/ibn/Desktop/report.txt";

fis = new FileInputStream(file);

status= client.storeFile(" /kk/"+file.getName(), fis);

client.logout();

fis.close();

}

catch(Exception e){

System.out.println(e);

}

if(status){

System.out.println("success");

return true;

}

else{

System.out.println("failed");

return false;

}

}

}

package algorithm;

import java.util.Properties;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class Mail {

public static boolean secretMail(String msg, String name, String email) {

Properties props = new Properties();

props.put("mail.smtp.host", "smtp.gmail.com");

props.put("mail.smtp.socketFactory.port", "465");

props.put("mail.smtp.socketFactory.class",

"javax.net.ssl.SSLSocketFactory");

props.put("mail.smtp.auth", "true");

props.put("mail.smtp.port", "465");

// Assuming you are sending email from localhost

Session session = Session.getDefaultInstance(props,

new javax.mail.Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication("jaswanthsai3459@gmail.com", "9676650989");

}

});

System.out.println("Message " + msg);

try {

Message message = new MimeMessage(session);

message.setFrom(new InternetAddress(name));

message.setRecipients(Message.RecipientType.TO,

InternetAddress.parse(email));

message.setSubject("Secret key");

message.setText(msg);

Transport.send(message);

System.out.println("Done");

return true;

} catch (MessagingException e) {

System.out.println(e);

e.printStackTrace();

return false;

// throw new RuntimeException(e);

}

}

}

package algorithm;

import java.io.InputStream;

import java.security.spec.KeySpec;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESedeKeySpec;

import org.apache.tomcat.util.codec.binary.Base64;

/\*\*

\*

\* @author java4

\*/

public class RS\_IBE {

private static final String UNICODE\_FORMAT = "UTF8";

public static final String DESEDE\_ENCRYPTION\_SCHEME = "DESede";

private KeySpec ks;

private SecretKeyFactory skf;

private Cipher cipher;

byte[] arrayBytes;

private String myEncryptionKey;

private String myEncryptionScheme;

SecretKey key;

public RS\_IBE() throws Exception {

myEncryptionKey = "ThisIsSpartaThisIsSparta";

myEncryptionScheme = DESEDE\_ENCRYPTION\_SCHEME;

arrayBytes = myEncryptionKey.getBytes(UNICODE\_FORMAT);

ks = new DESedeKeySpec(arrayBytes);

skf = SecretKeyFactory.getInstance(myEncryptionScheme);

cipher = Cipher.getInstance(myEncryptionScheme);

key = skf.generateSecret(ks);

}

public String encrypt(String unencryptedString) {

String encryptedString = null;

try {

cipher.init(Cipher.ENCRYPT\_MODE, key);

byte[] plainText = unencryptedString.getBytes(UNICODE\_FORMAT);

byte[] encryptedText = cipher.doFinal(plainText);

encryptedString = new String(Base64.encodeBase64(encryptedText));

} catch (Exception e) {

e.printStackTrace();

}

return encryptedString;

}

public String decrypt(String encryptedString) {

String decryptedText=null;

try {

cipher.init(Cipher.DECRYPT\_MODE, key);

byte[] encryptedText1 = encryptedString.getBytes(UNICODE\_FORMAT);

byte[] encryptedText = Base64.decodeBase64(encryptedText1);

byte[] plainText = cipher.doFinal(encryptedText);

decryptedText= new String(plainText);

} catch (Exception e) {

e.printStackTrace();

}

return decryptedText;

}

}

SOFTWARE ENVIRONMENT

CHAPTER 6

## SOFTWARE ENVIRONMENT

## Java Technology

Java technology is both a programming language and a platform.

### The Java Programming Language

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

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

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



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



### The Java Platform

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

The Java platform has two components:

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

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

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

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



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

## What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

* **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
* **Software components**: Known as JavaBeansTM, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



## How Will Java Technology Change My Life?

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

### ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or drivers. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

### JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. ***SQL Level API***

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

1. ***SQL Conformance***

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

1. ***JDBC must be implemental on top of common database interfaces***The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
2. ***Provide a Java interface that is consistent with the rest of the Java system***

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

1. ***Keep it simple***

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

1. ***Use strong, static typing wherever possible***

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

1. ***Keep the common cases simple***

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database.

Java ha two things: a programming language and a platform.

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

Simple Architecture-neutral

Object-oriented Portable

Distributed High-performance

Interpreted multithreaded

Robust Dynamic

Secure

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

## Networking

### TCP/IP stack

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



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

### IP datagram’s

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

### UDP

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

### TCP

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

### Internet addresses

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

### Network address

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

### Subnet address

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

### Host address

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

### Total address



The 32 bit address is usually written as 4 integers separated by dots.

### Port addresses

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

### Sockets

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

#include <sys/types.h>

#include <sys/socket.h>

int socket(int family, int type, int protocol);

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

**JFree Chart**

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

## 1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

## 2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

## 3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

## 4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

**J2ME (Java 2 Micro edition):-**

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

**1. General J2ME architecture**



J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

**2. Developing J2ME applications**

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

**3. Design considerations for small devices**

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

\* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

\* Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

\* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

**4. Configurations overview**

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

\* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

\* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

**5. J2ME profiles**

**What is a J2ME profile?**

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

**Profile 1: KJava**

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

**Profile 2: MIDP**

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

\* java.lang

\* java.io

\* java.util

\* javax.microedition.io

\* javax.microedition.lcdui

\* javax.microedition.midlet

\* javax.microedition.rms

SYSTEM TESTING

**CHAPTER 7**

**SYSTEM TESTING**

## SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

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

**Black Box Testing**

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

**6.1 Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**6.3 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**SYSTEM STUDY**

**8.SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# 

# 

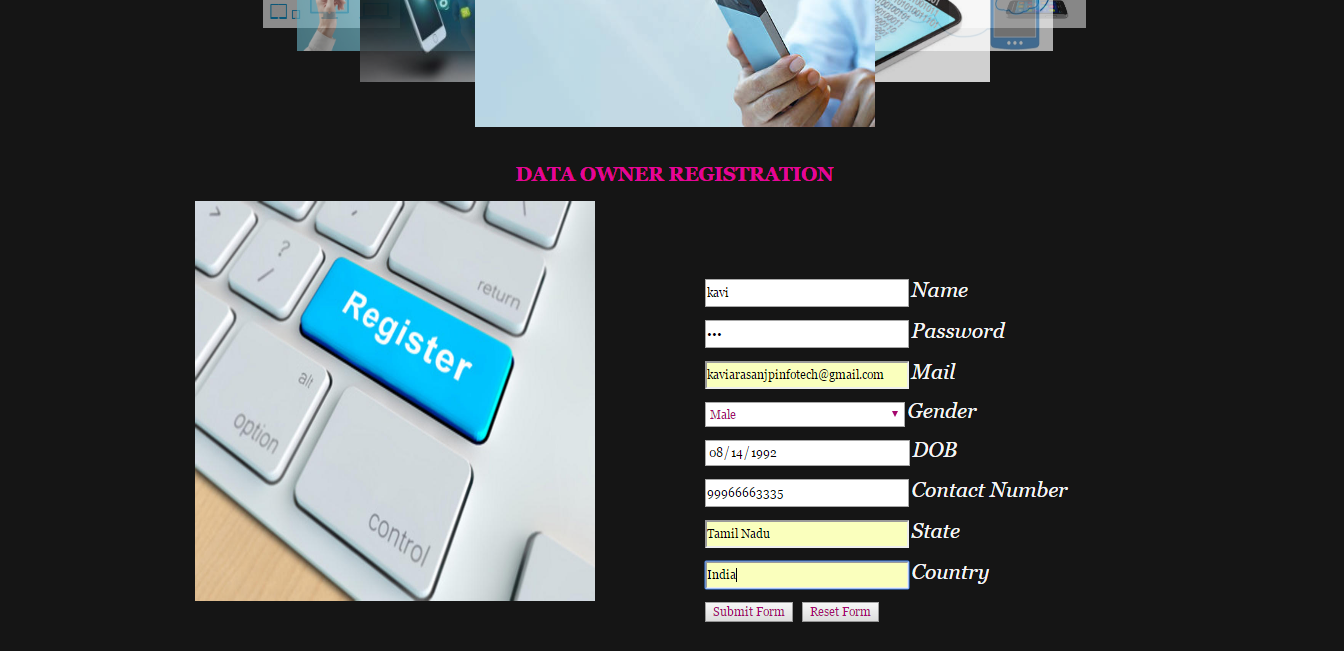
# OUTPUT SCREENSHOTS

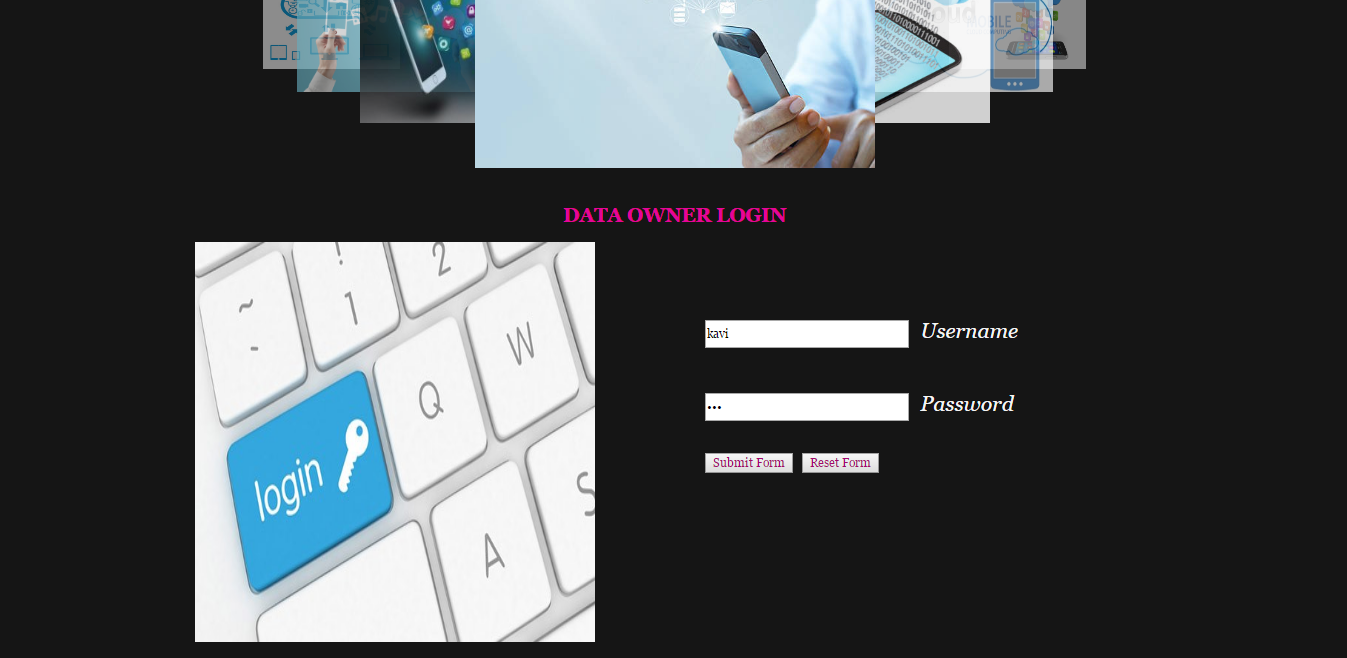
CHAPTER 8

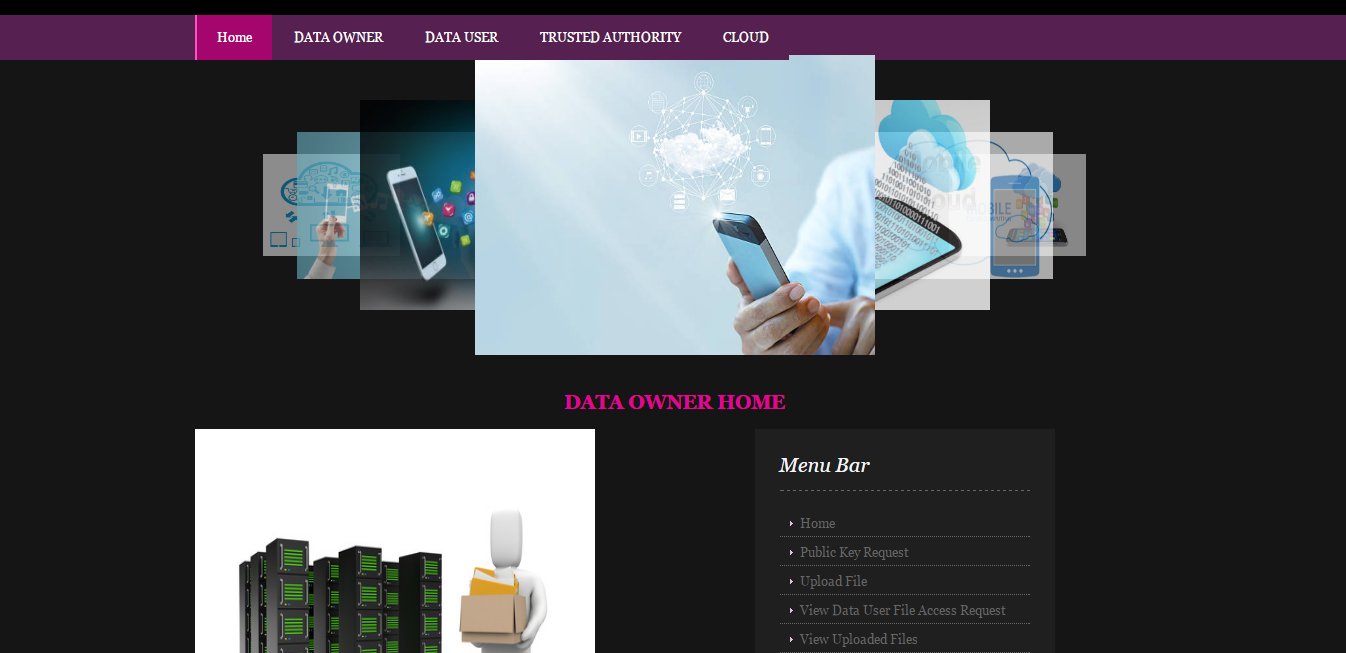
OUTPUT SCREENSHOTS

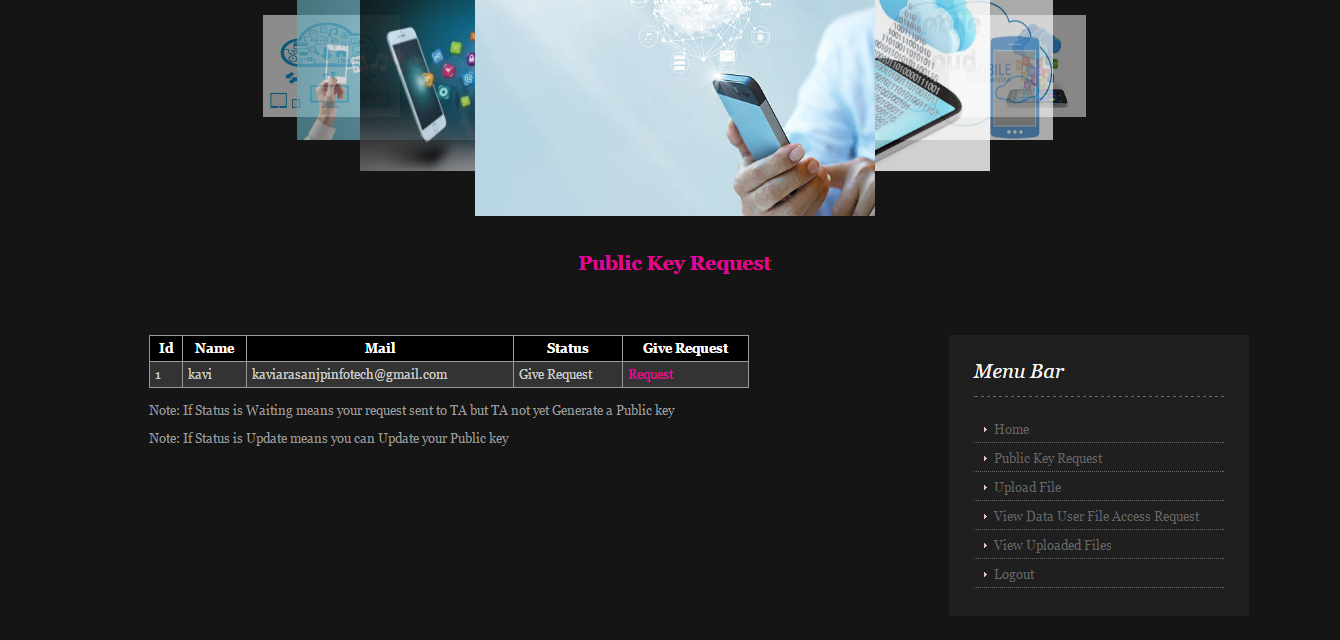
## OUTPUT SCREENSHOTS

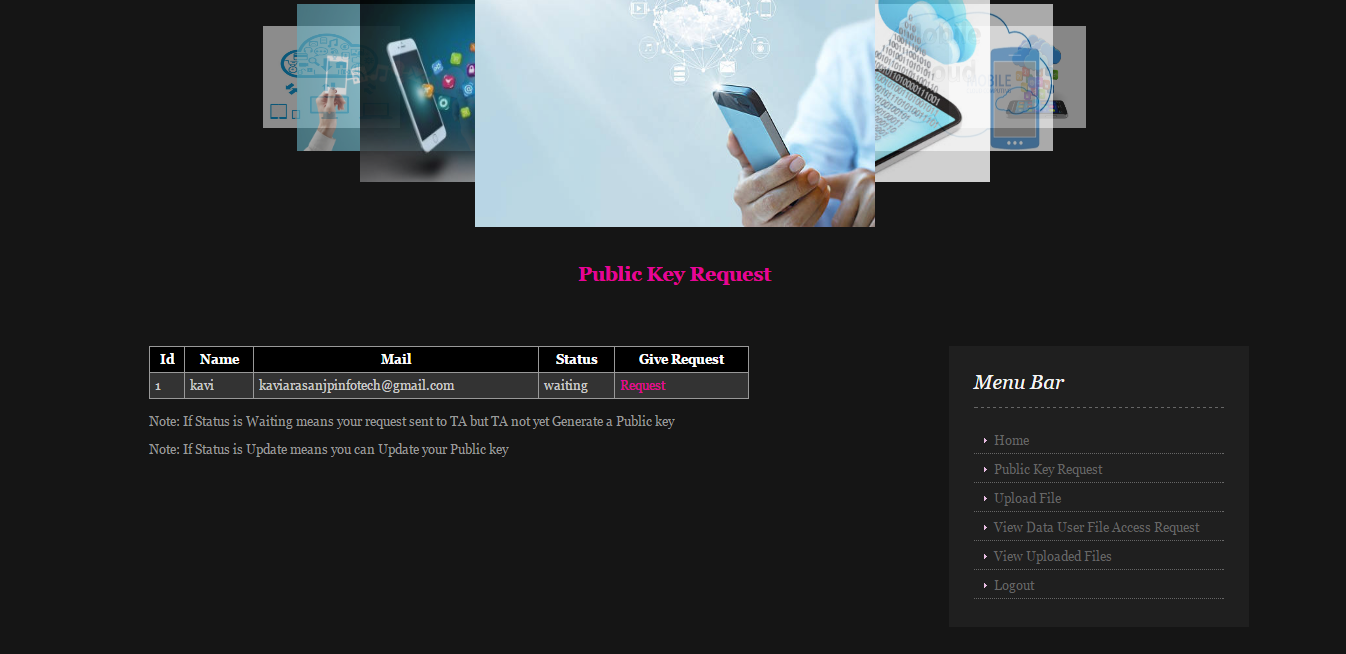












# 

# CONCLUSION

CHAPTER 9

CONCLUSION

## CONCLUSION

In recent years, many studies on access control in cloud are based on attribute-based encryption algorithm (ABE). However, traditional ABE is not suitable for mobile cloud because it is computationally intensive and mobile devices only have limited resources. In this paper, we propose LDSS to address this issue. It introduces a novel LDSS-CP-ABE algorithm to migrate major computation overhead from mobile devices onto proxy servers, thus it can solve the secure data sharing problem in mobile cloud. The experimental results show that LDSS can ensure data privacy in mobile cloud and reduce the overhead on users’ side in mobile cloud. In the future work, we will design new approaches to ensure data integrity. To further tap the potential of mobile cloud, we will also study how to do ciphertext retrieval over existing data sharing schemes.

# 

# BIBLIOGRAPHY

CHAPTER 10

BIBLIOGRAPHY

## REFERENCES

[1] Gentry C, Halevi S. Implementing gentry’s fully-homomorphic encryption scheme. in: Advances in Cryptology–EUROCRYPT 2011. Berlin, Heidelberg: Springer press, pp. 129-148, 2011.

[2] Brakerski Z, Vaikuntanathan V. Efficient fully homomorphic encryption from (standard) LWE. in: Proceeding of IEEE Symposium on Foundations of Computer Science. California, USA: IEEE press, pp. 97-106, Oct. 2011.

[3] Qihua Wang, Hongxia Jin. "Data leakage mitigation for discertionary access control in collaboration clouds". the 16th ACM Symposium on Access Control Models and Technologies (SACMAT), pp.103-122, Jun. 2011.

[4] Adam Skillen and Mohammad Mannan. On Implementing Deniable Storage Encryption for Mobile Devices. the 20th Annual Network and Distributed System Security Symposium (NDSS), Feb. 2013.

[5] Wang W, Li Z, Owens R, et al. Secure and efficient access to outsourced data. in: Proceedings of the 2009 ACM workshop on Cloud computing security. Chicago, USA: ACM pp. 55-66, 2009.

[6] Maheshwari U, Vingralek R, Shapiro W. How to build a trusted database system on untrusted storage. in: Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4. USENIX Association, pp. 10-12, 2000.

[7] Kan Yang, Xiaohua Jia, Kui Ren: Attribute-based fine-grained access control with efficient revocation in cloud storage systems. ASIACCS 2013, pp. 523-528, 2013.

[8] Crampton J, Martin K, Wild P. On key assignment for hierarchical access control. in: Computer Security Foundations Workshop. IEEE press, pp. 14-111, 2006.

[9] Shi E, Bethencourt J, Chan T H H, et al. Multi-dimensional range query over encrypted data. in: Proceedings of Symposium on Security and Privacy (SP), IEEE press, 2007. 350- 364

[10] Cong Wang, Kui Ren, Shucheng Yu, and Karthik Mahendra Raje Urs. Achieving Usable and Privacy-assured Similarity Search over Outsourced Cloud Data. IEEE INFOCOM 2012, Orlando, Florida, March 25-30, 2012

[11] Yu S., Wang C., Ren K., Lou W. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. INFOCOM 2010, pp. 534-542, 2010

[12] Kan Yang, Xiaohua Jia, Kui Ren, Bo Zhang, Ruitao Xie: DACMACS: Effective Data Access Control for Multiauthority Cloud Storage Systems. IEEE Transactions on Information Forensics and Security, Vol. 8, No. 11, pp.1790-1801, 2013.

[13] Stehlé D, Steinfeld R. Faster fully homomorphic encryption. in: Proceedings of 16th International Conference on the Theory and Application of Cryptology and Information Security. Singapore: Springer press, pp.377-394, 2010.

[14] Junzuo Lai, Robert H. Deng ,Yingjiu Li ,et al. Fully secure keypolicy attribute-based encryption with constant-size ciphertexts and fast decryption. In: Proceedings of the 9th ACM symposium on Information, Computer and Communications Security (ASIACCS), pp. 239-248, Jun. 2014.

[15] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute based encryption in: Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP). Washington, USA: IEEE Computer Society, pp. 321-334, 2007.

[16] Liang Xiaohui, Cao Zhenfu, Lin Huang, et al. Attribute based proxy re-encryption with delegating capabilities. in: Proceedings of the 4th International Symposium on Information, Computer and Communications Security. New

York, NY, USA: ACM press, pp. 276-286, 2009.

[17] Pirretti M, Traynor P, McDaniel P, et al. Secure atrribute-based systems. in: Proceedings of the 13th ACM Conference on Computer and Communications Security. New York, USA: ACM press, pp. 99-112, 2006.

[18] Yu S., Wang C., Ren K., et al. Attribute based data sharing with attribute revocation. in: Proceedings of the 5th International Symposium on Information, Computer and Communications Security (ASIACCS), New York, USA: ACM press pp. 261-270, 2010.

[19] Sandhu R S, Coyne E J, Feinstein H L, et al. Role-based access control models. Computer, 29(2): 38-47, 1996.

[20] Tian X X, Wang X L, Zhou A Y. DSP RE-Encryption: A flexible mechanism for access control enforcement management in DaaS. in: Proceedings of IEEE International Conference on Cloud Computing. IEEE press, pp.25-32, 2009

[21] Di Vimercati S D C, Foresti S, Jajodia S, et al. Over-encryption: management of access control evolution on outsourced data. in: Proceedings of the 33rd international conference on Very large data bases. Vienna, Austria: ACM, pp. 123-134, 2007.

[22] Kan Yang, Xiaohua Jia, Kui Ren, Ruitao Xie, Liusheng Huang: Enabling efficient access control with dynamic policy updating for big data in the cloud. INFOCOM 2014, pp.2013-2021, 2014.

[23] Jia W, Zhu H, Cao Z, et al. SDSM: a secure data service mechanism in mobile cloud computing. in: Proceedings of 30th IEEE International Conference on Computer Communications. Shanghai, China: IEEE, pp. 1060-1065, 2011.

[24] D. Huang, X. Zhang, M. Kang, and J. Luo. Mobicloud: A secure mobile cloud framework for pervasive mobile computing and communication. in: Proceedings of 5th IEEE International Symposium on Service-Oriented System Engineering. Nanjing, China: IEEE, pp. 90-98, 2010.

[25] Benjamin Livshits, Jaeyeon Jung. Automatic Mediation of Privacy-Sensitive Resource Access in Smartphone Applications. USENIX Security, pp.113-130, Aug. 2013.

[26] Zhou Z, Huang D. Efficient and secure data storage operations for mobile cloud computing. in: Proceedings of 8th International Conference on Network and Service Management (CNSM 2012), Las Vegas, USA: IEEE, pp. 37-45, 2012.

[27] P. K. Tysowski and M. A.Hasan. Hybrid attribute- and reencryption-based key management for secure and scalable mobile applications in clouds. IEEE Transactions on Cloud Computing, vol. 1, no. 2, pp. 172-186, Nov. 2013.

[28] Boneh D, Franklin M. Identity-based encryption from the Weil pairing. in: Proceedings of the Advances in Cryptology. Berlin, Heidelberg: Springer-Verlag, pp. 213−229, 2001.

[29] Sahai A, Waters B. Fuzzy identity based encryption. in: Proceedings of the Advances in Cryptology. Aarhus, Denmark: Springer-Verlag, pp.457-473, 2005.

[30] Shamir A. How to share a secret. Communications of the ACM,1979, 22 (11): 612-613.