**Towards Secure Data Distribution Systems In Mobile Cloud Computing**

**ABSTRACT**

Though the electronic technologies have passed through speedy developments in recent years, cellular gadgets such as smartphones are nonetheless relatively susceptible in evaluation to computers in phrases of computational capability, garage and so on, and aren't able to meet the growing demands from mobile users. By integrating cellular computing and cloud computing, cellular cloud computing (MCC) substantially extends the boundary of the cell programs, but it additionally inherits many challenges in cloud computing, e.G., facts privacy and records integrity. In this paper, we leverage numerous cryptographic primitives which include a new kind-based totally proxy re-encryption to layout a comfy and efficient statistics distribution device in MCC, which presents information privateness, facts integrity, information authentication, and flexible information distribution with get entry to manipulate. Compared to conventional cloud-based information garage systems, our gadget is a light-weight and without difficulty deployable solution for mobile users in MCC because no trusted 1/3 parties are involved and every cell user only has to hold brief mystery keys along with 3 group factors for all cryptographic operations. Finally, we present substantial overall performance analysis and empirical research to demonstrate the safety, scalability, and performance of our proposed system

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. OBJECTIVE   3. EXISTING SYSTEM   1.3.1 EXISTING SYSTEM DISADVANTAGES  1.3.2 LITERATURE SURVEY  1.4 PROPOSED SYSTEM  1.4.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 : PROJECT DESCRIPTION**  2.1 GENERAL  2.2 PROBLEM DEFINATION  2.3 METHODOLOGIES  2.3.1 MODULES NAME  2.3.2 MODULES EXPLANATION  2.3.3 MODULE DIAGRAM  2.3.4 GIVEN INPUTAND EXPECTED OUTPUT  2.4 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS  3.4 Functional Specification  3.5 Non-Functional SpecificatioN |  |
| 4. | **CHAPTER 4 : SYSTEM DESIGN**  4.1 general  4.1.1 USE CASE diagram  4.1.2 CLASS DIAGRAM  4.1.3 OBJECT DIAGRAM  4.1.4 STATE DIAGRAM  4.1.5 ACTIVITY DIAGRAM  4.1.6 SEQUENCED DIAGRAM  4.1.7 COLLABARATION DIAGRAM  4.1.8 COMPONENT DIAGRAM  4.1.9 DEPLOYMENT DIAGRAM  4.1.10 E-R DIAGRAM  4..2 DATA FLOW DIAGRAM  4.2.1 SYSTEM ARCHETECTURE |  |
| 5. | **CHAPTER 5 :SOFTWARE SPECIFICATION**  5.1 general FRONT END  5.2 features of java  5.2.1 the java framework  5.2.2 objective of java  5.2.3 componets of java frmework  5.2.4 Evolution of collection framework  5.3 features of the common language runtime  5.4 j2ee  5.4.1 EVALUTION OF J2EE  5.4.2 THE J2EE DATA ARCHITECTURE  5.5.3 COMPONENT CLASSES THAT MAKE UP  DATA PROVIDERS |  |
| 6. | **CHAPTER 6 : IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION  6.3 DATA BASE TABLE STRUCTURE |  |
| 7. | **CHAPTER 7 : SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 : SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 : APPLICATIONS AND FUTURE ENHANCEMENT**  8.1 GENERAL  8.2 APPLICATIONS  8.3 FUTURE ENHANCEMENTS |  |
| 10. | **CHAPTER 10 :**  9.1 CONCLUSION  9.2 REFERENCES |  |

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2 | Use case Diagram |  |
| 4.3 | Class Diagram |  |
| 4.4 | Object Diagram |  |
| 4.5 | State Diagram |  |
| 4.6 | Sequence Diagram |  |
| 4.7 | Collaboration Diagram |  |
| 4.8 | Activity Diagram |  |
| 4.9 | Component Diagram |  |
| 4.10 | Data flow Diagram |  |
| 4.11 | E-R Diagram |  |
| 4.12 | Deployment Diagram |  |
| 4.13 | System Architecture |  |
| 7.1 | Home Page |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **NAME** | **NOTATION** | | **DESCRIPTION** | |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | | Represents a collection of similar entities grouped together. | |
| 2. | Association | name  Class B  Class A  Class A  Class B | | Associations represents static relationships between classes. Roles represents the way the two classes see each other. | |
| 3. | Actor | Class A  Class A  Class B  Class B | | It aggregates several classes into a single classes. | |
| 5. | Aggregation | Interaction between the system and external environment | |
| 5. | Relation  (uses) | | Uses | | Used for additional process communication. | |
| 6. | Relation  (extends) | | extends | | Extends relationship is used when one use case is similar to another use case but does a bit more. | |
| 7. | Communication | |  | | Communication between various use cases. | |
| 8. | State | | State | | State of the process. | |
| 9. | Initial State | |  | | Initial state of the object | |
| 10. | Final state | |  | | Final state of the object | |
| 11. | Control flow | |  | | Represents various control flow between the states. | |
| 12. | Decision box | |  | | Represents decision making process from a constraint | |
| 13. | Usecase | |  | | Interact ion between the system and external environment. | |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | DataBase |
| 2. | JVM | Java Virtual Machine |
| 3. | JSP | Java Server Page |
| 4. | CB | Collective Behavior |
| **5.** | RSSS | Ramp secret sharing scheme |
| 6. | JRE | Java Runtime Environment |

**CHAPTER 1**

**INTRODUCTION**

**1.1 GENERAL:**

Nowadays, it becomes very common and popular to access cloud services by using mobile devices. By a recent study, cloud applications will account for 90% of total mobile data traffic by 2018. To offload storage to the cloud, there are many existing storage services for mobile devices, such as Dropbox, Box, iCloud, Google Drive, and Skydrive. Since mobile cloud computing (MCC) integrates mobile computing and cloud computing, all the above security issues in cloud computing are inherited in MCC with the extra resource limited mobile devices For example, in September 2014, a loophole in iCloud causes many private photos of iphone users to be downloaded by hackers. Thus, it is very necessary to design a framework to ensure the security of private data in cloud storages. Besides, the limitations of mobile devices such as low storage, unpredictable Internet connectivity and less energy also require a lightweight solution in MCC that provides security with minimum communication and processing overhead, which may make MCC services significantly different from cloud services for desktops. Recently, Huang et al. gave four principles for shifting mobile clouds from the traditional Internet clouds, where the first principle requires that MCC applications should be designed in such a way that a user can control their own data with strong privacy and security protection.

**1.2. OBJECTIVE**

Proposed concept deals with the concept of variations of the memory management. Proposed System deals with the mobile space to been connected with the cloud. This accessed space can been given to user for various transactions to keep their data safe and secure. The admin also checks the variation of file keys. In this work, we present an efficient data distribution system in MCC, which allows mobile users to securely store their data in the cloud storage, and flexibly share their data with friends.

* 1. **Existing System:**

Existing concept provide the low memory management. It tends to cause the attackers to attack the file easily. The Mobile has fixed Memory Transactions for storing the images,datas,files etc. The Retrival of Images can tends to upload the memory space.

**1.3.1 Existing System Disadvantages:**

* Low Memory Management.
* Wastage of Time.

**1.3.2 LITERATURE SURVEY**

**TITLE :** Provable data possession at un trusted stores

**AUTHOR :** Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring,

Lea Kissner, Zachary Peterson, and Dawn Song

**YEAR :** 2007

**DESCRIPTION**

We introduce a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage systems. We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation

**TITLE :** Improved proxy re-encryption schemes with applications to secure distributed storage.

**AUTHOR :** Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger

**YEAR :** 2006

**DESCRIPTION**

In 1998, Blaze, Bleumer, and Strauss (BBS) proposed an application called atomic proxy re-encryption, in which a semi-trusted proxy converts a ciphertext for Alice into a ciphertext for Bob without seeing the underlying plaintext. We predict that fast and secure re-encryption will become increasingly popular as a method for managing encrypted file systems. Although efficiently computable, the wide-spread adoption of BBS re-encryption has been hindered by considerable security risks. Following recent work of Dodis and Ivan, we present new re-encryption schemes that realize a stronger notion of security, and we demonstrate the usefulness of proxy re-encryption as a method of adding access control to a secure file system. Performance measurements of our experimental file system demonstrate that proxy re-encryption can work effectively in practice.

**TITLE :** An analysis of latent sector errors in disk drives

**AUTHOR :** Lakshmi N. Bairavasundaram, Garth R. Goodson, Shankar Pasupathy,

and Jiri Schindler

**YEAR :** 2007

**DESCRIPTION**

The reliability measures in today’s disk drive-based storage systems focus predominantly on protecting against complete disk failures. Previous disk reliability studies have analyzed empirical data in an attempt to better understand and predict disk failure rates. Yet, very little is known about the incidence of latent sector errors i.e., errors that go undetected until the corresponding disk sectors are accessed. Our study analyzes data collected from production storage systems over 32 months across 1.53 million disks (both nearline and enterprise class). We analyze factors that impact latent sector errors, observe trends, and explore their implications on the design of reliability mechanisms in storage systems. To the best of our knowledge, this is the first study of such large scale – our sample size is at least an order of magnitude larger than previously published studies – and the first one to focus specifically on latent sector errors and their implications on the design and reliability of storage systems.

**TITLE :** Enabling dynamic data and indirect mutual trust for cloud computing storage systems

**AUTHOR :** A. Barsoum and A. Hasan.

**YEAR :** 2013

**DESCRIPTION**

Storage-as-a-service offered by cloud service providers (CSPs) is a paid facility that enables organizations to outsource their sensitive data to be stored on remote servers. In this paper, we propose a cloud-based storage scheme that allows the data owner to benefit from the facilities offered by the CSP and enables indirect mutual trust between them. The proposed scheme has four important features: 1) it allows the owner to outsource sensitive data to a CSP, and perform full block-level dynamic operations on the outsourced data, i.e., block modification, insertion, deletion, and append, 2) it ensures that authorized users (i.e., those who have the right to access the owner's file) receive the latest version of the outsourced data, 3) it enables indirect mutual trust between the owner and the CSP, and 4) it allows the owner to grant or revoke access to the outsourced data. We discuss the security issues of the proposed scheme. Besides, we justify its performance through theoretical analysis and a prototype implementation on Amazon cloud platform to evaluate storage, communication, and computation overheads.

**TITLE :** Divertible protocols and atomic proxy cryptography

**AUTHOR :** Matt Blaze, Gerrit Bleumer, and Martin Strauss.

**YEAR :** 1998

**DESCRIPTION**

First, we introduce the notion of divertibility as a protocol property as opposed to the existing notion as a language property (see Okamoto, Ohta [OO90]). We give a definition of protocol divertibility that applies to arbitrary 2-party protocols and is compatible with Okamoto and Ohta's definition in the case of interactive zero-knowledge proofs. Other important examples falling under the new definition are blind signature protocols. We propose a sufficiency criterion for divertibility that is satisfied by many existing protocols and which, surprisingly, generalizes to cover several protocols not normally associated with divertibility (e.g., Diffie-Hellman key exchange). Next, we introduce atomic proxy cryptography, in which an atomic proxy function, in conjunction with a public proxy key, converts ciphertexts (messages or signatures) for one key into ciphertexts for another. Proxy keys, once generated, may be made public and proxy functions applied in untrusted environments. We present atomic proxy functions for discrete-log-based encryption, identification, and signature schemes. It is not clear whether atomic proxy functions exist in general for all public-key cryptosystems. Finally, we discuss the relationship between divertibility and proxy cryptography.

**TITLE :** Efficient selective identity-based encryption without random oracles

**AUTHOR :** Dan Boneh and Xavier Boyen.

**YEAR :** 2011

**DESCRIPTION**

We construct two efficient Identity-Based Encryption (IBE) systems that admit selectiveidentity security reductions without random oracles in groups equipped with a bilinear map. Selective-identity secure IBE is a slightly weaker security model than the standard security model for IBE. In this model the adversary must commit ahead of time to the identity that it intends to attack, whereas in an adaptive-identity attack the adversary is allowed to choose this identity adaptively. Our first system—BB1—is based on the well studied decisional bilinear Diffie-Hellman assumption, and extends naturally to systems with hierarchical identities, or HIBE. Our second system—BB2—is based on a stronger assumption which we call the Bilinear Diffie-Hellman Inversion assumption and provides another approach to building IBE systems. Our first system, BB1, is very versatile and well suited for practical applications: the basic hierarchical construction can be efficiently secured against chosen-ciphertext attacks, and further extended to support efficient non-interactive threshold decryption, among others, all without using random oracles. Both systems, BB1 and BB2, can be modified generically to provide “full” IBE security (i.e., against adaptive-identity attacks), either using random oracles, or in the standard model at the expense of a non-polynomial but easy-to-compensate security reduction.

* 1. **Proposed System**

In this work, we present an efficient data distribution system in MCC, which allows mobile users to securely store their data in the cloud storage, and flexibly share their data with friends. We leverage several cryptographic primitives to achieve data privacy, data integrity, dynamical data modification and deletion, and flexible data distribution. Concretely, we first design an efficient type-based proxy re-encryption (TB-PRE), which allows a mobile user with a single secret key to keep the data privacy, and flexibly share his data with friends under permission. We also use the BLS signature to both protect the integrity of the data and provide authentication to the data. By combining the BLS signature with Merkle hash tree (MHT), our system allows the data owner to dynamically modify and delete his data, and enables the data consumer to efficiently authenticate the data owner’s identity. In addition, our data distribution system is designed without any trusted third party (TTP) such that a mobile user can fully control his own data, and easily share it with friends. Actually, establishing a TTP might be a little troublesome when a smartphone user simply wants to share some pictures with his close friends.

**1.4.1 Proposed System Advantages**

* Low Memory Management.
* Wastage of Time.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

The data owner can arbitrarily classify his data into different categories and encrypt them under a single public key. The data owner can also check the integrity of the encrypted data, and dynamically modify or delete the encrypted data in the cloud. The data owner can permit data consumers to access his private data by the category of the data. More specifically, the data owner can allow different data consumers to access different set of data categories. The data consumer only has to interact with the data owner one time to get access permission for each data category. In other words, the data owner is not required to be online during the data distribution phase after granting the access permission. Any user without permission to access a particular data category of the data owner cannot read the data in the category, while a permitted data consumer can read the data, authenticate the data owner’s identity, and check the integrity of the data in the category. All the users only have to keep secret keys consisting of three group elements to perform all cryptographic operations, which is independent of the number of total users in the system. We give an extensive performance analysis and a proof-of-concept implementation to demonstrate the efficiency of our system in terms of storage, communication and computation overheads on both.

**2.3 METHODOLOGIES**

**2.3.1** **MODULES NAME:**

**This project having the following modules:**

* **User Interface**
* **Data Owner**
* **Data Consumer**
* **Secure Data Distribution and Integrity**
* **Summarization**
  + 1. **MODULE EXPLANATION**
* **User Interface:**

In this module we design windows are used for secure login for all users. To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download. Name will be set as user id. Logging in is usually used to enter a specific page.

* **Data Owner:**

In this module we can do data owner access to give all permissions of the data to all users from server credentials, here data owner interacts with data consumer on time to access the permission of the data to all the users. When users can interact with data owner data owner giving secure all operations like USER DETAILS, EDIT PROFILE, FILE UPLOAD,FILE DETAILS,FILE ACCESSING CONTROL, TRANSCATIONS….etc in data owner home page. This is how data owner give credentials to the users or data consumers.

* **Data Consumer:**

In this module mainly data consumer can act like new user to the data owner, once login credentials can be done data owner can give permissions to see data of data owner. Whatever the data he is sending to the server here mainly go with private key process of the data, based on data owner home data consumer can act actions like according to data owner home operations. This how data consumer retrieving form the data owner based on the secret key permissions.

* **Secure Data Distribution & Integrity:**

In this secure data distribution & integrity mainly based upon how the users can interact with the data owner and he/she enhancing the current operations the data consumer can send the data based upon the key generation technique. so that users can get more and more interest of securing data from data owner to data consumers.

* **Summarizations:**

In this project we are summarization mainly admin can to all operations of the data owner and data consumer. Here data consumers can interact with the data owner for retrieving the data operations and he/she make an events or operations to secure the data based the secret key generation of public and private data information’s.

**2.3.3 MODULE DIAGRAM:**

* **User Interface Design**

**Server**

**Register**

**Login**

**Cloud**

* **Data Owner Module:**

**Cloud Database**

**File upload, Download & Update**

**Verification of All authority**

**My rights & User verification**

**Owner page**

**Signature Key Verification**

**Login**

* **Data Consumer**

**Login**

**User Page**

**Verification of All authority**

**Download & Update**

**Cloud Database**

* **Secure Data Distribution and Integrity**

**Request to data owner**

**Encrypted file &key**

Request accepted

**Key for Decryption**

**Download File**

Clod Data base

* **Summarizations:**

Result Displayed

Admin

Login

Admin Page

Cloud Database

Display All DO Details

Display All User Details

data checking

**2.3.4 GIVEN INPUT EXPECTED OUTPUT:**

1. **User Interface Design:**

**Input:** User name and password

**Output:** User window

**2. Data Owner Module**

**Input:** Data Owner enter name and password & upload files

**Output:** Data Owner Login then upload files. This how data consumer retrieving form the data owner based on the secret key permissions.

1. **Data Consumer**

**Input:** Data consumer enter name & password

**Output:** User login then verify data . This how data consumer retrieving form the data owner based on the secret key permissions.

* **Secure Data Distribution and Integrity**

**Input:** In this secure data distribution & integrity mainly based.

**Output:** upon how the users can interact with the data owner and he/she enhancing the current operations the data consumer can send the data based upon the key generation technique. so that users can get more and more interest of securing data from data owner to data consumers.

1. **Summarizations:**

**Input:** Admin login, verify encryption data key and verify data.

**Output:** On clicking on each hyperlink he will be able to see what operations cloud users are doing in the cloud.

* 1. **TECHNIQUE USED OR ALGORITHM USED**
* **Web Service Setup Algorithm:**

This technique is used to link the mobile space with the cloud, that is used to store the unlimited data by means of flexibility We give an extensive performance analysis and a proof-of-concept implementation to demonstrate the efficiency of our system in terms of storage, communication and computation overheads on both sides of the cloud server and the mobile devices.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

In this paper, we proposed all these data anonymization mechanisms have serious side effects on the data utility. As a result, the users of the published data usually have a strong demand to verify the real utility of the anonymized data.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shouls what the system do and not how it should be implemented.

**HARDWARE**

* PROCESSOR : PENTIUM IV 2.6 GHz, Intel Core 2 Duo.
* RAM : 512 MB DD RAM
* MONITOR : 15” COLOR
* HARD DISK : 40 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* + Front End : J2EE (JSP, SERVLET)
  + Back End : MY SQL 5.5
  + Operating System : Windows 7
  + IDE : Eclipse

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, and outputs. Proposed concept deals with the concept of variations of the memory management. Proposed System deals with the mobile space to been connected with the cloud. This accessed space can been given to user for various transactions to keep their data safe and secure. The admin also checks the variation of file keys.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

* The data owner can arbitrarily classify his data into different categories and encrypt them under a single public key. The data owner can also check the integrity of the encrypted data, and dynamically modify or delete the encrypted data in the cloud.
* The data owner can permit data consumers to access his private data by the category of the data. More specifically, the data owner can allow different data consumers to access different set of data categories.
* The data consumer only has to interact with the data owner one time to get access permission for each data category. In other words, the data owner is not required to be online during the data distribution phase after granting the access permission.
* Any user without permission to access a particular data category of the data owner cannot read the data in the category, while a permitted data consumer can read the data, authenticate the data owner’s identity, and check the integrity of the data in the category.
* All the users only have to keep secret keys consisting of three group elements to perform all cryptographic operations, which is independent of the number of total users in the system.
* We give an extensive performance analysis and a proof-of-concept implementation to demonstrate the efficiency of our system in terms of storage, communication and computation overheads on both.

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.2 Use Case Diagram**



**EXPLANATION:**

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. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**4.3 Class Diagram**



**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project

**4.4 Object Diagram**



**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**4.5 State Chart Diagram**



**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.6 Sequence Diagram**



**EXPLANATION:**

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. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

**4.7 Collaboration Diagram**

****

**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

**4.8 Activity Diagram**



**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**4.9 Component Diagram**



**EXPLANATION:**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

**4.10 Data Flow Diagram:**

**Level 0:**

Login Page

User Page

Distribution Data Integrity

Verify All Access Permissions

File Upload & Download

Cloud Database

**Level 1:**

Key Generation

Login

User Page

Secret Code Encrypt/Decrypt Download

Cloud Database

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**4.11 E-R Diagram:**

**Data Owner**

**Cloud Authority**

**Data Consumer**

**Verifies**

**Verifies**

**Verifies**

**Login**

**Response for request**

**Request for file**

**Encryption key**

**Upload file**

**Verify key**

**Download file**

**Cloud Database**

**Accept request**

**EXPLANATION:**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**4.12 Deployment Diagram:**



**4.13 System Architecture**

DATA OWNER

Owner

Login

Key Verification

User Page

All Authority

File Upload, Download & Update

DATA CONSUMER

User Page

Login

All Access Permissions

Download, Update & View Data

Cloud Server

Encryption Process

Decryption Process

**EXPLANATION:**

Data consumer can act like new user to the data owner, once login credentials can be done data owner can give permissions to see data of data owner. Whatever the data he is sending to the server here mainly go with private key process of the data, based on data owner home data consumer can act actions like according to data owner home operations. This how data consumer retrieving form the data owner based on the secret key permissions.

**CHAPTER 5**

**DEVELOPMENT TOOLS**

* 1. **GENERAL**

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA. The Primary languages are JAVA, J2EE and J2ME. In this project J2EE is chosen for implementation.

**5.2 FEATURES OF JAVA**

**5.2.1 THE JAVA FRAMEWORK**

**Java** is a programming language originally developed by James Gosling at Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications the java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal The data owner is a mobile user who stores his

private data in the cloud, and allows the data consumer to access his private data from the cloud. The cloud is an entity who provides storage services and is responsible to help the data owner to distribute the private data to the data consumer. The data consumer is an entity who first gets data access permission from the data owner, and then access the data owner’s private data from the cloud.

**5.2.2 OBJECTIVES OF JAVA**

To see places of Java in Action in our daily life, explore java.com.

## Why Software Developers Choose Java

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

* Write software on one platform and run it on virtually any other platform
* Create programs to run within a Web browser and Web services
* Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
* Combine applications or services using the Java language to create highly customized applications or services
* Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

## Some Ways Software Developers Learn Java

Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

**ObjectOriented** To be an Object Oriented language, any language must follow at least the four characteristics.

1. Inheritance   :It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse  the existing code and adding addition a features as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the  methods.

4. Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

**5.2.3 JAVA SWING OVERVIEW**

**Abstract Window Toolkit (AWT) is cross-platform**

Swing[[1]](http://en.wikibooks.org/wiki/Java_Programming/Swing#cite_note-0) provides many controls and widgets to build user interfaces with. Swing class names typically begin with a J such as JButton, JList, JFrame. This is mainly to differentiate them from their AWT counterparts and in general is one-to-one replacements. Swing is built on the concept of Lightweight components vs AWT and SWT's concept of Heavyweight components. The difference between the two is that the Lightweight components are rendered (drawn) using purely Java code, such as drawLine and drawImage, whereas Heavyweight components use the native operating system to render the components.

Some components in Swing are actually heavyweight components. The top-level classes and any derived from them are heavyweight as they extend the AWT versions. This is needed because at the root of the UI, the parent windows need to be provided by the OS. These top-level classes include JWindow, JFrame, JDialog and JApplet. All Swing components to be rendered to the screen must be able to trace their way to a root window of one of those classes.

**Note**: It generally it is not a good idea to mix heavyweight components with lightweight components (other than as previously mentioned) as you will encounter layering issues, e.g., a lightweight component that should appear "on top" ends up being obscured by a heavyweight component. The few exceptions to this include using heavyweight components as the root pane and for popup windows. Generally speaking, heavyweight components will render on top of lightweight components and will not be consistent with the look and feel being used in Swing. There are exceptions, but that is an advanced topic. The truly adventurous may want to consider reading this [article](http://java.sun.com/products/jfc/tsc/articles/mixing/) from Sun on mixing heavyweight and lightweight components.

**5.2.4 Evolution of Collection Framework:**

Almost all collections in Java are derived from the [**java.util.Collection**](http://download.oracle.com/javase/7/docs/api/java/util/Collection.html) interface. Collection defines the basic parts of all collections. The interface states the add() and remove() methods for adding to and removing from a collection respectively. Also required is the toArray() method, which converts the collection into a simple array of all the elements in the collection. Finally, the contains() method checks if a specified element is in the collection. The Collection interface is a subinterface of [**java.util.Iterable**](http://download.oracle.com/javase/7/docs/api/java/util/Iterable.html), so the iterator() method is also provided. All collections have an iterator that goes through all of the elements in the collection. Additionally, Collection is a generic. Any collection can be written to store any class. For example, Collection<String> can hold strings, and the elements from the collection can be used as strings without any casting required.

There are three main types of collections:

* Lists: always ordered, may contain duplicates and can be handled the same way as usual arrays
* Sets: cannot contain duplicates and provide random access to their elements
* Maps: connect unique keys with values, provide random access to its keys and may host duplicate values

**LIST**

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two concrete classes implement List. The first is java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it. The other implementation is java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.

SET:

Java's [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet,java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a [java.util.HashMap](http://download.oracle.com/javase/7/docs/api/java/util/HashMap.html) to store the hashes and elements and to prevent duplicates. Java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. [java.util.TreeSet](http://download.oracle.com/javase/7/docs/api/java/util/TreeSet.html) uses a red-black tree implemented by a [java.util.TreeMap](http://download.oracle.com/javase/7/docs/api/java/util/TreeMap.html). The red-black tree makes sure that there are no duplicates. Additionally, it allows Tree Set to implement java.util.SortedSet.

The [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method, or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet

[java.util.SortedSet](http://download.oracle.com/javase/7/docs/api/java/util/SortedSet.html) is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.

**MAP:**

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it's just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

**Thread:**

Simply put, a threadis a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time.

## Creating threads

Java's creators have graciously designed two ways of creating threads: implementing an interface and extending a class. Extending a class is the way Java inherits methods and variables from a parent class. In this case, one can only extend or inherit from a single parent class. This limitation within Java can be overcome by implementing interfaces, which is the most common way to create threads. (Note that the act of inheriting merely allows the class to be run as a thread. It is up to the class to start() execution, etc.)

Interfaces provide a way for programmers to lay the groundwork of a class. They are used to design the requirements for a set of classes to implement. The interface sets everything up, and the class or classes that implement the interface do all the work. The different set of classes that implement the interface have to follow the same rules.

**5.5 Conclusion**

Swing's high level of flexibility is reflected in its inherent ability to override the native host [operating system](http://en.wikipedia.org/wiki/Operating_system) (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. The Java thread scheduler is very simple. All threads have a priority value which can be changed dynamically by calls to the threads setPriority() method . Implementing the above concepts in our project to do the efficient work among the Server.

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding:**

**Ulogin.jsp**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns=*"http://www.w3.org/1999/xhtml"*>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"* />

<!--<link href="style.css" rel="stylesheet" type="text/css" />-->

<script type=*"text/javascript"* src=*"js/jquery.js"*></script>

<script type=*"text/javascript"* src=*"js/cufon-yui.js"*></script>

<script type=*"text/javascript"* src=*"js/arial.js"*></script>

<script type=*"text/javascript"* src=*"js/cuf\_run.js"*></script>

<script type=*"text/javascript"* src=*"js/radius.js"*></script><link rel=*"stylesheet"* href=*"http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css"* />

<script src=*"http://code.jquery.com/jquery-1.9.1.js"*></script>

<script src=*"http://code.jquery.com/ui/1.10.3/jquery-ui.js"*></script>

<!--<link rel="stylesheet" href="/resources/demos/style.css" />-->

</head>

<body background=*"images/bgimg3.jpg"*>

<center> <img src=*"images/title.png"* align=*"middle"*/> </center>

<a href=*"HomePage.jsp"*>Home</a>

<br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/><br/>

<center>

<form name=*"f"* action=*"lohinAction"* method=*"post"* onSubmit="return valid();">

<h3> <marquee scrollamount=*"5"* width=*"40"*>&lt;&lt;&lt;</marquee>User Login<marquee scrollamount=*"5"* direction=*"right"* width=*"40"*>&gt;&gt;&gt;</marquee></h3>

<br />Username: <input type=*"text"* id=*"user"* name=*"username"* title=*"Username"* value=*""* size=*"30"* maxlength=*"2048"* /><br /><br />

Password: <input name=*"password"* type=*"password"* id=*"pass"* title=*"Password"* value=*""* size=*"30"* maxlength=*"2048"* /><br /><br />

<input type=*"submit"* value=*"Submit"*/> <input type=*"reset"* value=*"Reset"*/><br /><br />

<a href=*"userRegister.jsp"*>New User ?</a>

</form></center>

<br/><br/><br/><br/><br/> <br/><br/><br/><br/><br/>

<p>Copyright &copy;<a href=*"http://www.vertilinktech.com/"*> Vertilink.</a> All Rights Reserved. Design idea by<a href=*"#"*> Kishan </a></p>

</body>

</html>

**lohinAction.java**

package actionS;

import Utils.DbConnector;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class lohinAction extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

try { String v1 = request.getParameter("username");

String v2 = request.getParameter("password");

System.out.println(v1+","+v2);

Connection con = DbConnector.getConnection();

Statement st = con.createStatement();

String selec = "select \* from userreg where name = '"+v1+"' ";

ResultSet rs = st.executeQuery(selec);

if(rs.next()){

String duser = rs.getString("name");

String dpass = rs.getString("pass");

if(v1.equals(duser)&&(v2.equals(dpass))){

// response.sendRedirect("userHome.jsp?m="+v1);

response.sendRedirect("userLoginChk.jsp?m="+v1);

} else{ response.sendRedirect("login.jsp?m=login failed...!");

} }

if(v1.equalsIgnoreCase("ADMIN")&&(v2.equalsIgnoreCase("ADMIN"))){

response.sendRedirect("AdminPage.jsp");

} } catch (SQLException ex) {

Logger.getLogger(lohinAction.class.getName()).log(Level.SEVERE, null, ex);

} finally {

out.println("<h2 style='margin-left:1s0px;margin-top:250px;'> <center>UserName / PassWord Error.. <br> Check The Fileds.....!</center></h2>");

// response.sendRedirect("login.jsp?m=check Filedz");

} }

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

public String getServletInfo() {

return "Short description";

}// </editor-fold> }

**adminUpdate.java**

package actionS;

import Utils.DbConnector;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class adminUpdate extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

try { String mail = request.getParameter("mail");

String mob = request.getParameter("mob");

String addr = request.getParameter("addr");

Connection con = DbConnector.getConnection();

Statement st = con.createStatement();

int d = st.executeUpdate("update adminreg set email = '"+mail+"',mobile = '"+mob+"', addr = '"+addr+"' where name= '"+request.getParameter("name") +"' ");

response.sendRedirect("AdminPage.jsp?l = profile Updated..!");

} catch (SQLException ex) {

Logger.getLogger(adminUpdate.class.getName()).log(Level.SEVERE, null, ex);

} finally {

out.close(); } }

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**CHK.java**

package actionS;

import Utils.DbConnector;

import java.io.IOException;

import java.sql.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.\*;

public class CHK extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

try { String name=null;

Connection con = DbConnector.getConnection();

Statement st = con.createStatement();

ResultSet rs = st.executeQuery("select \* from userreg where name ='"+request.getParameter("userid") +"'");

if(rs.next()){ name=rs.getString(1);

HttpSession h=request.getSession();

h.setAttribute("name", name);

if(request.getParameter("key").equals(rs.getString("key"))){

java.util.Date dt=new java.util.Date();

PreparedStatement ps=con.prepareStatement("UPDATE userreg SET DATE='"+dt+"' where name ='"+request.getParameter("userid") +"'");

ps.executeUpdate();

response.sendRedirect("userHome.jsp");

} else{

response.sendRedirect("Ulogin.jsp?pp=check your key");

} } else{

response.sendRedirect("Ulogin.jsp?pp=check your key");

} } catch (SQLException ex) {

Logger.getLogger(CHK.class.getName()).log(Level.SEVERE, null, ex);

} finally { } }

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

processRequest(request, response);

} @Override

public String getServletInfo() {

return "Short description";

}// </editor-fold>

}

**CHAPTER 7**

**SNAPSHOTS**

**General:**

**This project is implements like web application using COREJAVA and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.**

**SNAPSHOTS**

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

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.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3 Types of Tests**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input 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.

**8.3.2 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.

**8.3.3 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.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 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.

**8.3.6 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.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**APPLICATION**

**9.1 GENERAL**

The above security challenges in MCC and the current state of cloud storage systems such as suggest that it is not an easy task to design an efficient and secure data distribution system in MCC. However, we observe that most mobile devices are currently in personal use, which usually does not need a data distribution system as powerful as that for business desktops. For example, in contrast to enterprise users in cloud storage who have to share hundreds of gigabytes of data among thousands of enterprise employees, a smartphone user usually shares many megabytes of photos with small number of friends. This fact makes it possible to design a lightweight data distribution system. In addition, as the developments of electronic technologies, the capability of mobile devices is greatly enhanced. One should not overlook the tradeoff between the energy consumption on the mobile devices and the expense of using cloud resources while designing a security framework.

**FUTURE ENHANCEMENT**

Modern operating systems perform sequential prefetching of file data from disk, also known as read-ahead, to reduce data access latency for future file operations. Given the widespread adoption of read-ahead, eviction algorithms of hypervisor based. On a guest OS’s put page request, the page is added to Synergy cache irrespective of whether it can be shared or not. The page remains a candidate for sharing (if an opportunity arises in the future) until it is evicted from the cache. Similarly it provide multiple keyword to check the file stored in cloud.

**CHAPTER 10**

**CONCLUSION & REFERENCE**

**10.1 CONCLUSION**

We propose a practical data distribution system in mobile cloud computing, which does not involve any trusted third party and provides several useful properties including data privacy, data integrity, data authentication, dynamic data modifications and deletions, as well as fine-grained access control. Our system leverages a new efficient and provably secure type-based proxy re-encryption scheme, Merkle hash tree, as well as the BLS signature to ensure the security. An extensive performance analysis and a proof-of-concept implementation show that our data distribution is practical.

**10.2 REFERENCE:**

[1] AmazonS3, <http://aws.amazon.com/s3/>.

[2] Cisco visual networking index: Global mobile data traffic forecast update, 2013–2018, February 5, 2014. <http://www>. cisco.com/c/en/us/solutions/collateral/service-provider/ visual-networking-index-vni/white paper c11-520862.html.

[3] MIRACL Crypto SDK, <https://certivox.com/>.

[4] Secure Hash Standard (SHS), National Institute of Standards and Technology (NIST), FIPS PUB 180-4, http://csrc.nist.gov/publications/PubsFIPS.html.

[5] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS ’07, pages 598–609, New York, NY, USA, 2007. ACM.

[6] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur., 9:1–30, February 2006.

[7] Lakshmi N. Bairavasundaram, Garth R. Goodson, Shankar Pasupathy, and Jiri Schindler. An analysis of latent sector errors in disk drives. In Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS ’07, pages 289–300, New York, NY, USA, 2007. ACM.

[8] PauloS.L.M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, Selected Areas in Cryptography, volume 3897 of Lecture Notes in Computer Science, pages 319–331. Springer Berlin Heidelberg, 2006.

[9] A. Barsoum and A. Hasan. Enabling dynamic data and indirect mutual trust for cloud computing storage systems. Parallel and Distributed Systems, IEEE Transactions on, 24(12):2375–2385, Dec 2013.

[10] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, Advances in Cryptology - EUROCRYPT’98, volume 1403 of LNCS, pages 127–144. Springer Berlin / Heidelberg, 1998.

[11] Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. Journal of Cryptology (JoC), 24(4):659–693, 2011. early version in Eurocrypt 2004.

[12] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In Colin Boyd, editor, Advances in Cryptology – ASIACRYPT 2001, volume 2248 of Lecture Notes in Computer Science, pages 514–532. Springer Berlin Heidelberg, 2001.

[13] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Proceedings of the 14th ACM conference on Computer and communications security, CCS ’07, pages 185–194, New York, NY, USA, 2007. ACM.

[14] Ee-Chien Chang and Jia Xu. Remote integrity check with dishonest storage server. In Sushil Jajodia and Javier Lopez, editors, Computer Security - ESORICS 2008, volume 5283 of Lecture Notes in Computer Science, pages 223–237. Springer Berlin Heidelberg, 2008.

[15] Melissa Chase and Sherman S.M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS ’09, pages 121–130, New York, NY, USA, 2009. ACM.

[16] Cheng-Kang Chu and Wen-Guey Tzeng. Identity-based proxy re-encryption without random oracles. In Juan Garay, Arjen Lenstra, Masahiro Mambo, and Ren Peralta, editors, Information Security, volume 4779 of LNCS, pages 189–202. Springer Berlin / Heidelberg, 2007.

[17] Adrian Covert. “Google Drive, iCloud, Dropbox and more compared: Whats the best cloud option?”, 2012. <http://gizmodo.com/> 5904739.

[18] Chris Erway, Alptekin K¨upc¸ ¨ u, Charalampos Papamanthou, and Roberto Tamassia. Dynamic provable data possession. In Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS ’09, pages 213–222, New York, NY, USA, 2009. ACM.

[19] Amos Fiat and Moni Naor. Broadcast encryption. In DouglasR. Stinson, editor, Advances in Cryptology ł CRYPTO 93, volume 773 of Lecture Notes in Computer Science, pages 480–491. Springer Berlin Heidelberg, 1994.

[20] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications security, CCS ’06, pages 89–98, New York, NY, USA, 2006. ACM.

[21] Matthew Green and Giuseppe Ateniese. Identity-based proxy reencryption. In Jonathan Katz and Moti Yung, editors, Applied Cryptography and Network Security - ACNS 2007, volume 4521 of LNCS, pages 288–306. Springer Berlin / Heidelberg, 2007.

[22] Dijiang Huang, Tianyi Xing, and Huijun Wu. Mobile cloud computing service models: a user-centric approach. Network, IEEE, 27(5):6–11, September 2013.

[23] Nguyen Thanh Hung, Do Hoang Giang, Ng Wee Keong, and Huafei Zhu. Cloud-enabled data sharing model. In Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on, pages 1–6, 2012.

[24] Junbeom Hur and Dong Kun Noh. Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Transactions on Parallel and Distributed Systems, 22(7):1214–1221, 2011.

[25] F. R. Institute. Personal data in the cloud: A global survey of consumer attitudes [online], 2010. Available:http://www.fujitsu.com/downloads/SOL/fai/reports/fujitsu personal-data-in-the-cloud.pdf.

[26] Ari Juels and Burton S. Kaliski, Jr. Pors: Proofs of retrievability for large files. In Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS ’07, pages 584–597, New York, NY, USA, 2007. ACM.

[27] Ryan Kalember. “celebrity photo hack: Ten takeaways for enterprises”, September 11, 2014. https://www.watchdox.com/en/ blog/celebrity-ph oto-hack-ten-takeaways-enterprises/.

[28] Abdul Nasir Khan, M.L. Mat Kiah, Samee U. Khan, and Sajjad A. Madani. Towards secure mobile cloud computing: A survey. Future Generation Computer Systems, 29(5):1278 – 1299, 2013. Special section: Hybrid Cloud Computing.

[29] Eike Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In Tatsuaki Okamoto and Xiaoyun Wang, editors, Public Key Cryptography – PKC 2007, volume 4450 of Lecture Notes in Computer Science, pages 282–297. Springer Berlin Heidelberg, 2007.

[30] Jason Kincaid. “Google privacy blunder shares your docs without permission”, 2009. <http://techcrunch.com/2009/03/07/> huge-google-privacy-blunder-shares-your-docs-without-permission.

[31] KMPG. From hype to future: Kpmgs 2010 cloud computing survey [online], 2010. Available:http://www.kpmg.com/ES/es/ActualidadyNovedades/ArticulosyPublicaciones/Documents/2010-Cloud-Computing-Survey.pdf.

[32] Noam Kogan, Yuval Shavitt, and Avishai Wool. A practical revocation scheme for broadcast encryption using smartcards. ACM Trans. Inf. Syst. Secur., 9(3):325–351, August 2006.

[33] Karthik Kumar and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? Computer, 43(4):51–56, 2010.

[34] Ming Li, Shucheng Yu, Yao Zheng, Kui Ren, and Wenjing Lou. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. Parallel and Distributed Systems, IEEE Transactions on, 24(1):131–143, Jan 2013.

[35] Benoˆıt Libert and Damien Vergnaud. Unidirectional chosenciphertext secure proxy re-encryption. Information Theory, IEEE Transactions on, 57(3):1786 –1802, march 2011.

[36] Qin Liu, Guojun Wang, and Jie Wu. Clock-based proxy reencryption scheme in unreliable clouds. In Parallel Processing Workshops (ICPPW), 2012 41st International Conference on, pages 304–305, 2012.

[37] Xuefeng Liu, Yuqing Zhang, BoyangWang, and Jingbo Yan. Mona: Secure multi-owner data sharing for dynamic groups in the cloud. IEEE Transactions on Parallel and Distributed Systems, 24(6):1182– 1191, 2013.

[38] Robert McMillan. “Hacker: I broke into twitter”, 2009. http:// www.pcworld.com/article/164182.

[39] Ralph C. Merkle. Protocols for public key cryptosystems. In IEEE Symposium on Security and Privacy. IEEE Computer Society, Los Alamitos, CA, USA, 1980.

[40] M. Nabeel, Ning Shang, and E. Bertino. Privacy preserving policy-based content sharing in public clouds. Knowledge and Data Engineering, IEEE Transactions on, 25(11):2602–2614, Nov 2013.