A Preliminary Report On

A Novel Secure Cloud Data Under Key Exposure

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

M.YOGANANDA REDDY

REG.NO.179Y1A0589

I.RAM BHUPAL REDDY

REG.NO.179Y1A0556

K. VENKATA SUBBA REDDY

REG.NO.179Y1A0565

K. VENKATA SAIKUMAR REDDY

REG.NO: 179Y1A0562

Under the Esteemed Guidance of

Dr. V. LOKESWARA REDDY M.Tech, Ph.D.
Professor
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
K.S.R.M COLLEGE OF ENGINEERING (AUTONOMOUS)
(Approved by AICTE, New Delhi & Affiliated to J.N.T.U.A., Ananthapuramu)
(Accredited by NAAC, Bangalore)
(An ISO 9001:2015 Certified Institution)
KADAPA - 516 004(A.P)

2020 - 2021

Vision and Mission of the Computer Science and Engineering Department

Vision:

To offer advanced subjects in a flexible curriculum which will evolve our graduates to be competent, responding successfully to career opportunities to meet the ongoing needs of the industry.

To progress as a centre of excellence adapting itself to the rapid developments in the field of computer science by performing a high-impact research and teaching environment.

Mission:

1. To impart high quality professional training in postgraduate and undergraduate level with strong emphasis on basic principles of Computer Science and Engineering.
2. To provide our students state-of-the-art academic environment and make unceasing attempts to instil the values that will prepare them for continuous learning.
3. To empower the youth in surrounding rural area with basics of computer education making them self-sufficient individuals.
4. To create teaching-learning environment that emphasizes depth, originality and critical thinking fostering leading-edge research in the ever-changing field of computer science.

K.S.R.M COLLEGE OF ENGINEERING (AUTONOMOUS)

(Approved by AICTE, New Delhi & Affiliated to J.N.T.U.A., Ananthapuramu)
(Accredited by NAAC, Bangalore)
(An ISO 9001:2015 Certified Institution)
KADAPA - 516 004(A.P)
2020 - 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to Certify that the Project work entitled "**A Novel Secure Cloud Data Under Key Exposure**" is work done by " **M. Yogananda Reddy(179Y1A0589), I. Ram Bhupal Reddy(179Y1A0556), K. Venkata Saikumar Reddy(179y1a0562), K. Venkata Subba Reddy (179y1a0565)**", being submitted to Jawaharlal Nehru Technological University Anantapur, ANANTAPURAMU in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in "**COMPUTER SCIENCE AND ENGINEERING**" during the period 2020- 2021

Project Guide:	Head of Department:
Dr. V. LOKESWARA REDDY M.Tech , Ph.D.	Dr.M. SRINIVASULU M.E, Ph.D.
Professor	Professor & HOD
Department of CSE	Department of CSE

Date: Internal Examiner External Examiner

ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to all those who gave us the possibility to complete this project work.

We would like to express our sincere gratitude and deep respect to our project supervisor **Dr**. **V. LOKESWARA REDDY MTech, Ph.D.,** Professor, Department of Computer Science and Engineering, for his encouragement, excellent guidance and kind support. We are thankful to him for being highly cooperative throughout our project work.

We are extremely grateful to **Dr.M. SRINIVASULU**, **M.E**, **Ph.D**., Head of the Department of Computer Science and Engineering, for his encouragement and support throughout the project work.

Special thanks are dedicated to **Dr. V.S.S. Murthy Ph.D., Principal, K.S.R.M. College of Engineering** who helped us for the successful completion of our project work.

We are extremely grateful to Prof. **A. Mohan M.Tech.,** Director, Kandula Group of Institutions who encouraged us and helped us for the successful completion of our project work.

It is our pleasure to express thanks to all the teaching and non-teaching staff of Computer Science and Engineering Department for their excellent monitoring and their suggestions. Finally, we thank our parents and the Almighty, whose divine grace provided us the opportunity to do our project work according to our wish.

Project Associates:

M.YOGANANDA REDDY (179Y1A0589)

K. VENKATA SAIKUMAR REDDY (179Y1A0562)

I.RAM BHUPAL REDDY (179Y1A0556)

K. VENKATA SUBBA REDDY (179Y1A0565)

CONTENTS

LIST OF FIGURES

LIST OF SCREENS

LIST OF ABBREVIATION

ABBREVIATION	FULL FORM
JSP	Java Server Page
SQL	Structured Query Language
GU	Graphical User Interface
UML	Unified Modeling Language
JVM	Java Virtual Machine
API	Application Programming Interface
HTML	Hyper Text Markup Language
JDBC	Java Database Connectivity
CSP	Cloud Service Provider

ABSTRACT

Recently reveals a powerful attacker which breaks data confidentiality by acquiring cryptographic keys. Once the encryption key is exposed, the only viable measure to preserve data confidentiality is to limit the attacker's access to the ciphertext. This may be achieved by spreading ciphertext blocks across servers in multiple administrative domains ,thus assuming that the adversary cannot compromise all of them. Nevertheless, if data is encrypted with existing schemes, an adversary equipped with the encryption key can still compromise a single server and decrypt the ciphertext blocks stored therein. In this project, we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the ciphertext blocks. In this project, we propose Bastion, a novel and efficient scheme that guarantees data confidentiality even if the encryption key is leaked and the adversary has access to almost all ciphertext blocks.

1. INTRODUCTION

1.1. ABOUT THE PROJECT

The world recently witnessed a massive surveillance program aimed at breaking users' privacy. Perpetrators were not hindered by the various security measures deployed within the targeted services. For instance, although these services relied on encryption mechanisms to guarantee data confidentiality, the necessary keying material was acquired by means of backdoors, bribes, or coercion. If the encryption key is exposed, the only viable means to guarantee confidentiality is to limit the adversary's access to the ciphertext, e.g., by spreading it across multiple administrative domains, in the hope that the adversary cannot compromise all of them. However, even if the data is encrypted and dispersed across different administrative domains, an adversary equipped with the appropriate keying material can compromise a server in one domain and decrypt ciphertext blocks stored. The adversary can acquire the key either by exploiting flaws or backdoors in the key-generation software, or by compromising the devices that store the keys (e.g., at the user-side or in the cloud). As far as we are aware, this adversary invalidates the security of most cryptographic solutions, including those that protect encryption keys by means of secret-sharing (since these keys can be leaked as soon as they are generated). To counter such an adversary, we propose Bastion, a novel and efficient scheme which ensures that plaintext data cannot be recovered as long as the adversary has access to at most all but two ciphertext blocks, even when the encryption key is exposed. Bastion achieves this by combining the use of standard encryption functions with an efficient linear transform. In this sense, Bastion shares similarities with the notion of all-or-nothing transform. An AONT is not an encryption by itself, but can be used as a pre-processing step before encrypting the data with a block cipher. This encryption paradigm called AON encryption was mainly intended to slow down brute-force attacks on the encryption key. However, AON encryption can also preserve data confidentiality in case the encryption key is exposed, as long as the adversary has access to at most all but one ciphertext block.

1.2. PROJECT DESCRIPTION

In this project,Once the encryption key is exposed, the only viable measure to preserve data confidentiality is to limit the attacker's access to the ciphertext. This may be achieved by spreading ciphertext blocks across servers in multiple administrative domains thus assuming that the adversary cannot compromise all of them. Nevertheless, if data is encrypted with existing schemes, an adversary equipped with the encryption key can still compromise a single server and decrypt the ciphertext blocks stored therein. In this project, we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the ciphertext blocks. In this project, we propose Bastion, a novel and efficient scheme that guarantees data confidentiality even if the encryption key is leaked and the adversary has access to almost all ciphertext blocks.

1.3. MODULES OF THE PROJECT

1.3.1 User Module:

The data consumer (User) is assigned a global user identity Urid. The user possesses a set of attributes and is equipped with a secret key associated with his/her attribute set.

The user can freely get any interested encrypted data from the cloud server. However, the user can decrypt the encrypted data if and only if his/her attribute set satisfies the access policy embedded in the encrypted data.

1.3.2 Owner Module:

The data owner defines the access policy about who can get access to each file, and encrypts the file under the defined policy. First of all, each owner encrypts his/her data with a symmetric encryption algorithm(AES) .

Then, the owner formulates access policy over an attribute set and encrypts the symmetric key under the policy according to public keys obtained from AA.

After that, the owner sends the whole encrypted data and the encrypted symmetric key (denoted as ciphertext CT) to the cloud server to be stored in the cloud.

1.3.3. Admin Module:

Admin is a super user, they can view all the user and owner details. admin can view the chart based on the greatest number of word searches, they can add related words, so users can easily map related words.

For example, Ambiguity level 2 refers to instances that most people think of as ambiguous.

1.3.4 Time based Key Module:

A time-based key generator is well secured for cloud storage data, because it's dynamic, based on time it will be generated so hackers aren't able to break the data usually, this process is fully used for security purposes.

1.3.5 Chart Module:

Chart Module based on number of file downloads in particular users, central authority can easily find out which file will be downloaded more. Here we are using a bar chart, scatter chart.

ARCHITECTURE DIAGRAM:

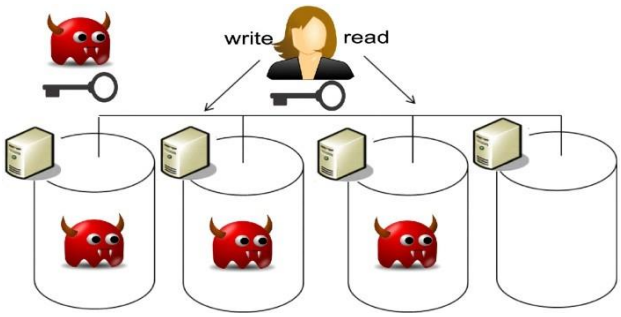


Fig 1.1. System Architecture

2. SYSTEM ANALYSIS

Analysis is a logical process. The objective of this phase is to determine exactly what must be done to solve the problem. Tools such as Class Diagrams, Sequence Diagrams, data flow diagrams and data dictionaries are used in developing a logical model of a system.

2.1. DOMAIN ANALYSIS

Domain analysis is the process by which a software engineer learns background information, which helps to understand the problem. The word ‘domain’ in the case means the general field of business or technology in which the customers expect to be using the software.

For this project, personal experiences of the team members with competing software were observed to understand the domain.

2.2. REQUIREMENT ANALYSIS

A requirement is a relatively short and concise piece of information, expressed as a fact. It can be written as a sentence or can be expressed using some kind of diagram.

2.2.1. Functional Requirements

Functional requirements describe what the system should do. The functional requirements can be further categorized as follows:

1. What inputs should the system accept?
2. What outputs should the system produce?
3. What data the system must store?
4. What are the computations to be done?

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and the steps are necessary to put transaction data into a usable form for processing that can be achieved by inspecting the computer to read data from a written or

printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining privacy. Input Design considered the following things:

1. What data should be given as input?
2. How should the data be arranged or coded?
3. The dialog to guide the operating personnel in providing input.
4. Methods for preparing input validations and steps to follow when errors occur.

2.2.2. Non-Functional Requirements

Non-functional requirements are the constraints that must be adhered to during development. They limit what resources can be used and set bounds on aspects of the software's quality.

2.2.3. User Interfaces

The User Interface is a GUI developed using Java.

2.2.4. Software Interfaces

The main processing is done in Java and console applications.

2.2.5. Manpower Requirements

Members can complete the project in 2 – 4 months if they work fulltime on it.

2.3. EXISTING SYSTEM

Cloud computing is internet-based computing which enables sharing of services. Many users place their data in the cloud. However, the fact that users no longer have physical possession of the possibly large size of outsourced data makes the data integrity protection in cloud computing a very challenging and potentially formidable task, especially for users with constrained computing resources and capabilities. So, correctness of data and security is a prime concern. This article studies the problem of ensuring the integrity and security of data storage in Cloud Computing. Security in the cloud is achieved by signing the data block before sending it to the cloud. Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in Cloud Computing a formidable task, especially for users with constrained computing resources.

2.3.1. Disadvantages

- The existing ABE schemes based on AND-Gate with wildcards cannot achieve this property.
- The existing ABE schemes based on AND-Gate with wildcards cannot achieve this. ABE can well protect the secrecy of the encrypted data against unauthorized access, it does not protect the privacy of the receivers/descriptors by default. That is, given the ciphertext, an unauthorized user may still be able to obtain some information from the data recipients.

2.4. PROPOSED SYSTEM

The world recently witnessed a massive surveillance program aimed at breaking users' privacy. Perpetrators were not hindered by the various security measures deployed within the targeted services. For instance, although these services relied on encryption mechanisms to guarantee data confidentiality, the necessary keying material was acquired by means of backdoors, bribes, or coercion. If the encryption key is exposed, the only viable means to guarantee confidentiality is to limit the adversary’s access to the ciphertext, e.g., by spreading it across multiple administrative domains, in the hope that the adversary cannot compromise all of them. However, even if the data is encrypted and dispersed across different administrative domains, an adversary equipped with the appropriate keying material can compromise a server in one domain and decrypt ciphertext blocks stored therein. In this project, we study data confidentiality against an adversary which knows the encryption key and has access to a large fraction of the ciphertext blocks. The

adversary can acquire the key either by exploiting flaws or 4 backdoors in the key-generation software, or by compromising the devices that store the keys (e.g., at the user-side or in the cloud). As far as we are aware, this adversary invalidates the security of most.

2.4.1 Disadvantages

- 0ur new technique leads to a new CP-ABE scheme with constant ciphertext size
- The system was used in the first construction to bridge ABE based on AND-Gate with a wildcard with Inner Product Encryption (IPE).
- Our first scheme achieves constant ciphertext size.
- Secure under the Decisional Bilinear Diffie-Hellman and the Decision Linear assumptions.

3. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and a 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.

3.1. ECONOMICAL FEASIBILITY

The economic issues usually arise during the economic feasibility stage are whether the system will be used if it is developed and implemented, whether the financial benefits are equal and exceeds the costs. The cost for developing the project will include the cost of conducting a full system investigation, cost of hardware and software for Three Level Password System Dept. of CSE, KSRMCE Page 9 the class being considered, the benefits in the form of reduced costs or fewer costly errors. The project is economically feasible if it is developed and installed. It reduces the workload. Keep the class of application in the view, the cost of hardware and software is considered to be economically feasible.

3.2. TECHNICAL FEASIBILITY

This application is going to be used in an internet environment called WWW (World Wide Web). So, it is necessary to use a technology that is capable of providing the networking facility to the application. This application is also able to work in a distributed environment. Application developed with JAVA Technology. One major advantage in application is platform neutral. We can deploy and use it in any operating system.

3.3 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.

4.SYSTEM REQUIREMENTS

4.1. SOFTWARE REQUIREMENTS

Operating System	:	Window 7/8/10
Coding Language	:	Java 1.8 and J2EE
Fronted Technologies	:	Html, JavaScript, CSS
Bank end Technologies	:	JSP, JDBC and Servlets
IDE tool	:	My Eclipse
Web Server	:	Apache Tomcat
Database	:	My SQL 5.1.44
Java Version	:	J2SDK1.8

4.2. HARDWARE REQUIREMENTS

Processor	:	Intel Core i3
Speed	:	1.3GHz
RAM	:	4GB
Hard Disk	:	500GB

4. SYSTEM DESIGN

5.1. UML DIAGRAM'S INTRODUCTION

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntax, semantic and pragmatic rules. A UML system is represented using five different views that describe the system from a distinctly different perspective.

UML is specifically constructed through two different domains they are:

- UML Analysis modeling, this focuses on the user model and structural model views of the system.
- UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

5.2. SYSTEM DESIGN ASPECTS

Once the analysis stage is completed, the next stage is to determine in broad outline form how the problem might be solved. During system design, we are beginning to move from the logical to physical level.

System design involves architectural and detailed design of the system. Architectural design involves identifying software components, decomposing them into processing modules and conceptual data structures, and specifying the interconnections among components.

Detailed design is concerned with how to package processing modules and how to implement the processing algorithms, data structures and interconnections of standard algorithms, invention of new algorithms, and design of data representations and packaging of software product**s.** Two kinds of approaches are available**:**

- Top-down approach
- Bottom-up approach

5.2.1. Design of Code

. Design of Code Since information systems projects are designed with space, time and cost saving in mind, coding methods in which conditions, words, ideas or control errors and speed the entire process. The purpose of the code is to facilitate the identification and retrieval of the information. A code is an ordered collection of symbols designed to provide unique identification of an entity or an attribute.

5.2.2. Design of Input

Design of input involves the following decisions

- Input data
- Input medium
- The way data should be arranged or coded
- Validation needed to detect every step to follow when error occurs

The input controls provide ways to ensure that only authorized users access the system guarantee the valid transactions, validate the data for accuracy and determine whether any necessary data has been omitted. The primary input medium chosen is display. Screens have been developed for input of data using HTML. The validations for all important inputs are taken care of through various events using JSP control.

5.2.3. Design of Output

Design of output involves the following decisions

- Information to present
- Output medium
- Output layout

Output of this system is given in an easily understandable, user-friendly manner, Layout of the output is decided through the discussions with the different users.

5.2.4 Design of Control

The system should offer the means of detecting and handling errors. Input controls provides ways per

- Valid transactions are only acceptable
- Validates the accuracy of data
- Ensures that all mandatory data have been captured

All entities to the system will be validated. And updating of tables is allowed for only valid entries. Means have been provided to correct, if any by change incorrect entries have been entered into the system they can be edited.

5.3. UML DIAGRAMS

Why Do We Use UML in projects?

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. Additionally, the development for the World Wide Web, while making some things simpler, has exacerbated these architectural problems. The Unified Modeling Language (UML) was designed to respond to these needs. Simply, Systems design refers to the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements which can be done easily through UML diagrams.

In the project four basic UML diagrams have been explained among the following list:

- Class Diagram
- Use Case Diagram
- Sequence Diagram
- Activity Diagram
- Collaboration Diagram
- Deployment Diagram
- State Chart Diagram
- Component Diagram

Class Diagram

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, and the relationships between the classes.

This is one of the most important of the diagrams in development. The diagram breaks the class into three layers. One has the name, the second describes its attributes and the third its methods. A padlock to the left of the name represents the private attributes. The relationships are drawn between the classes. Developers use the Class Diagram to develop the classes. Analyses use it to show the details of the system.

Architects look at class diagrams to see if any class has too many functions and see if they are required to be split.

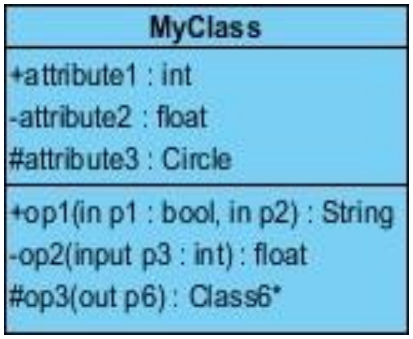


Fig.5.1: Class Diagram

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. Use cases are used during requirements elicitation and analysis to represent the functionality of the system.. Use cases focus on the behavior of the system from the external point of view. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of thesystem.

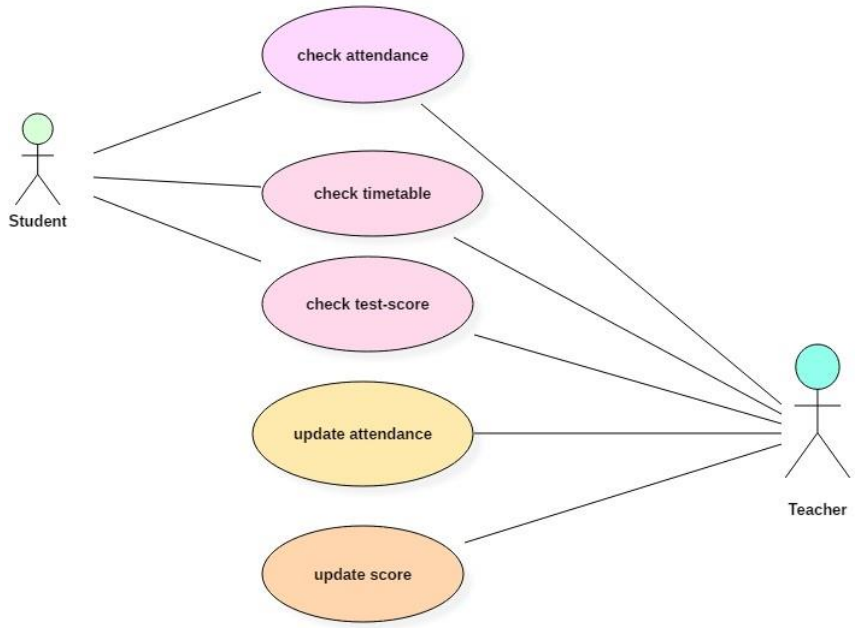


Fig.5.2: Use Case Diagram

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-trace diagrams, event scenarios, and timing diagrams

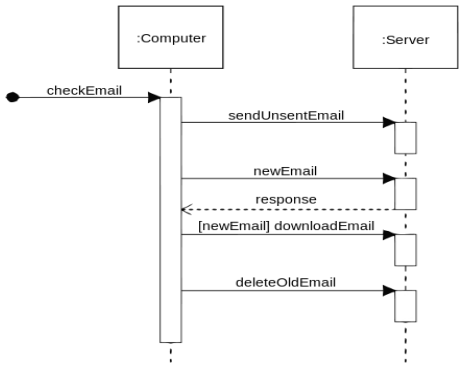


Fig.5.3: Sequence Diagram

Activity Diagram

Activity diagrams are a loosely defined diagram technique for showing 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.

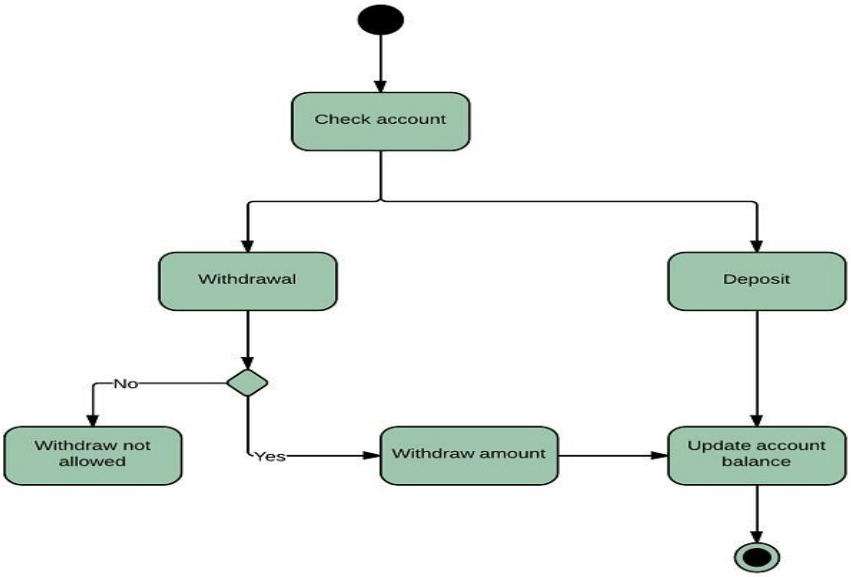


Fig.5.4: Activity Diagram

Collaboration Diagram

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.

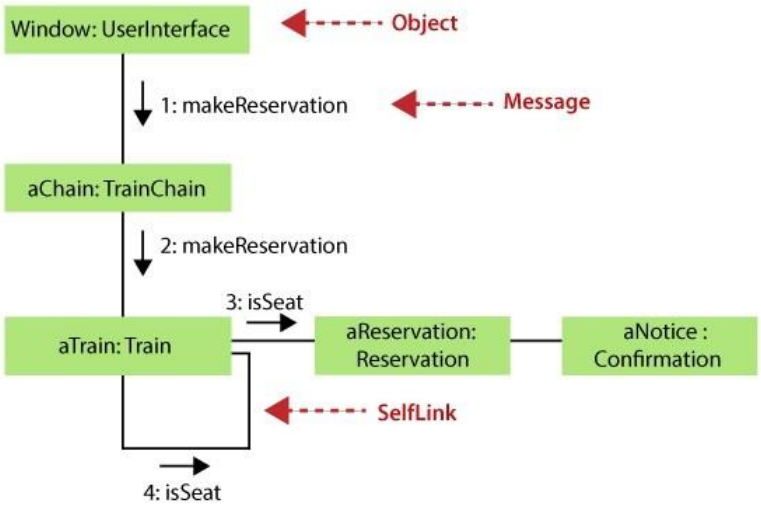


Fig.5.5: Collaboration Diagram

Deployment Diagram

A Deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected e.g. JDBC, REST

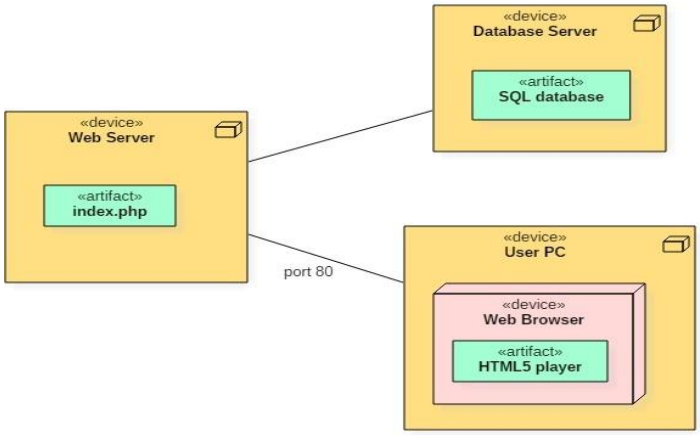


Fig.5.6: Deployment Diagram

State Chart Diagram

A State diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. 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

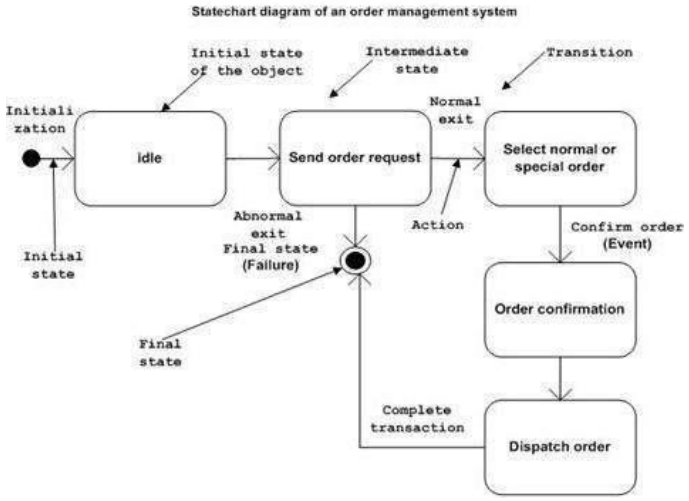


Fig.5.7: State Chart Diagram

Component Diagram

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.

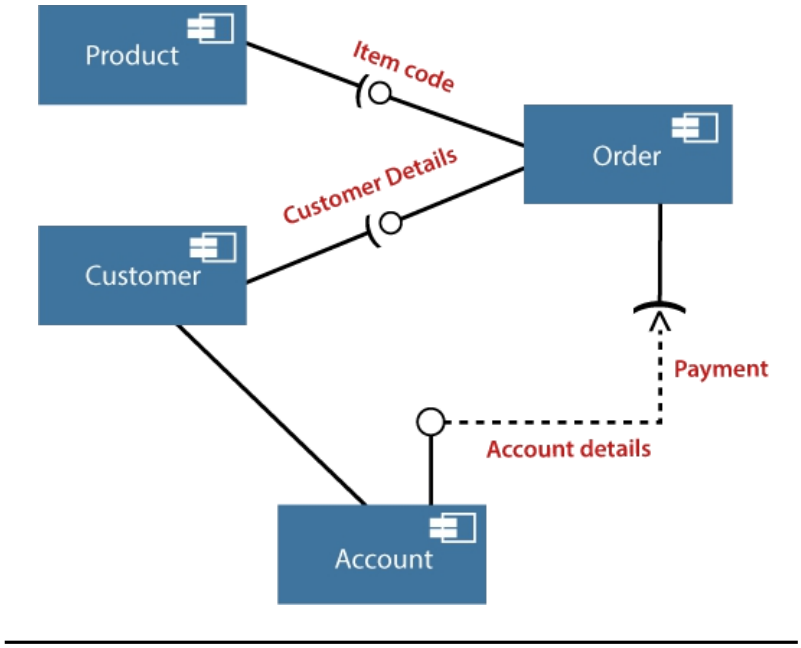


Fig.5.8: Component Diagram

DATA FLOW DIAGRAM

1. The DFD is also called a 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.

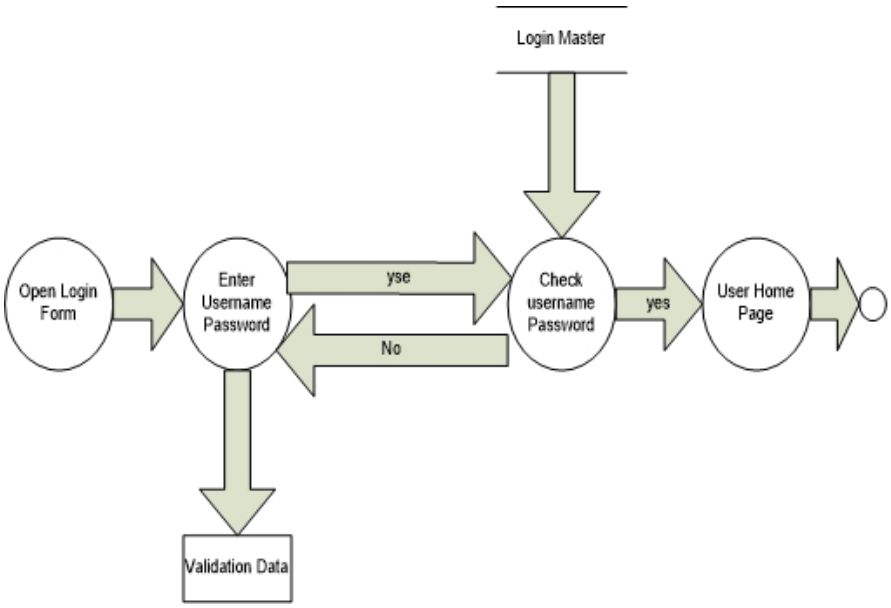


fig 5.9. Data Flow Diagram

UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML comprises two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

Class Diagram

The class diagram depicts a static view of an application. It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code.

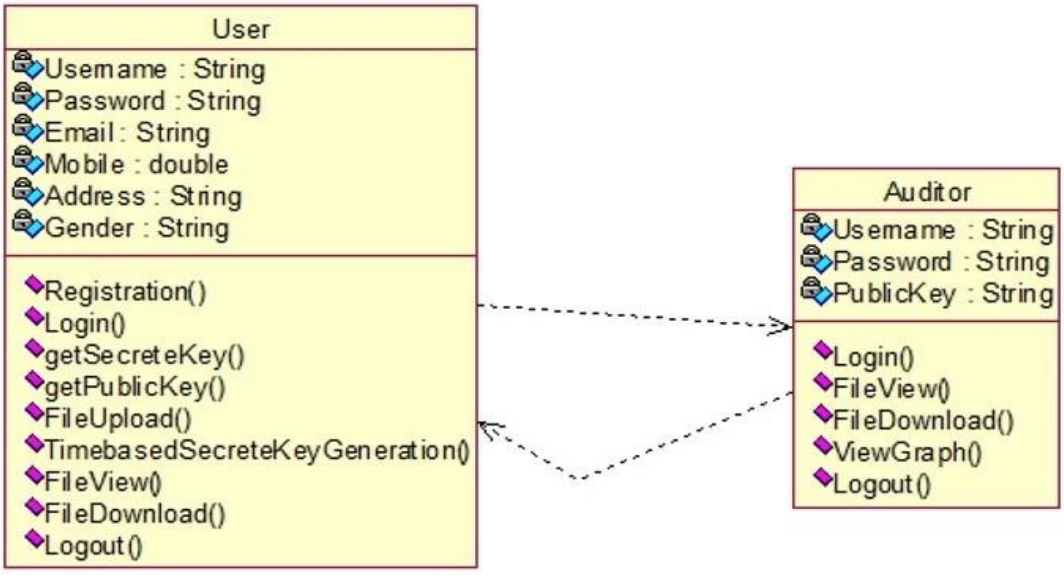


Fig .Class Diagram

5.3.1. Activity Diagram User

An activity diagram shows the flow from activity to activity. The activity diagram emphasizes the dynamic view of a system. It consists of activity states, action states, transition, and objects.

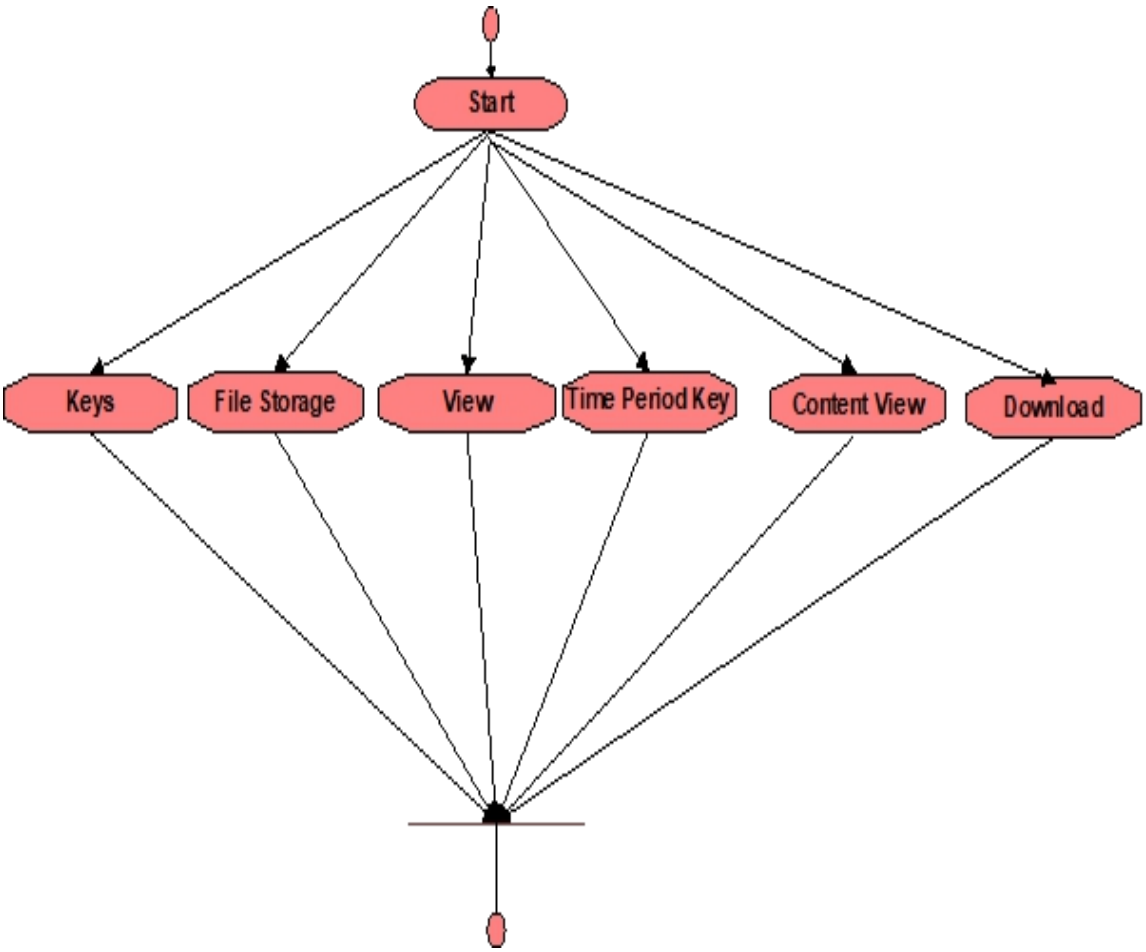


Fig 5.10 Activity Diagram User

5.3.2. Activity Diagram Auditor

An activity diagram shows the flow from activity to activity. The activity diagram emphasizes the dynamic view of a system. It consists of activity states, action states, transition, and objects.

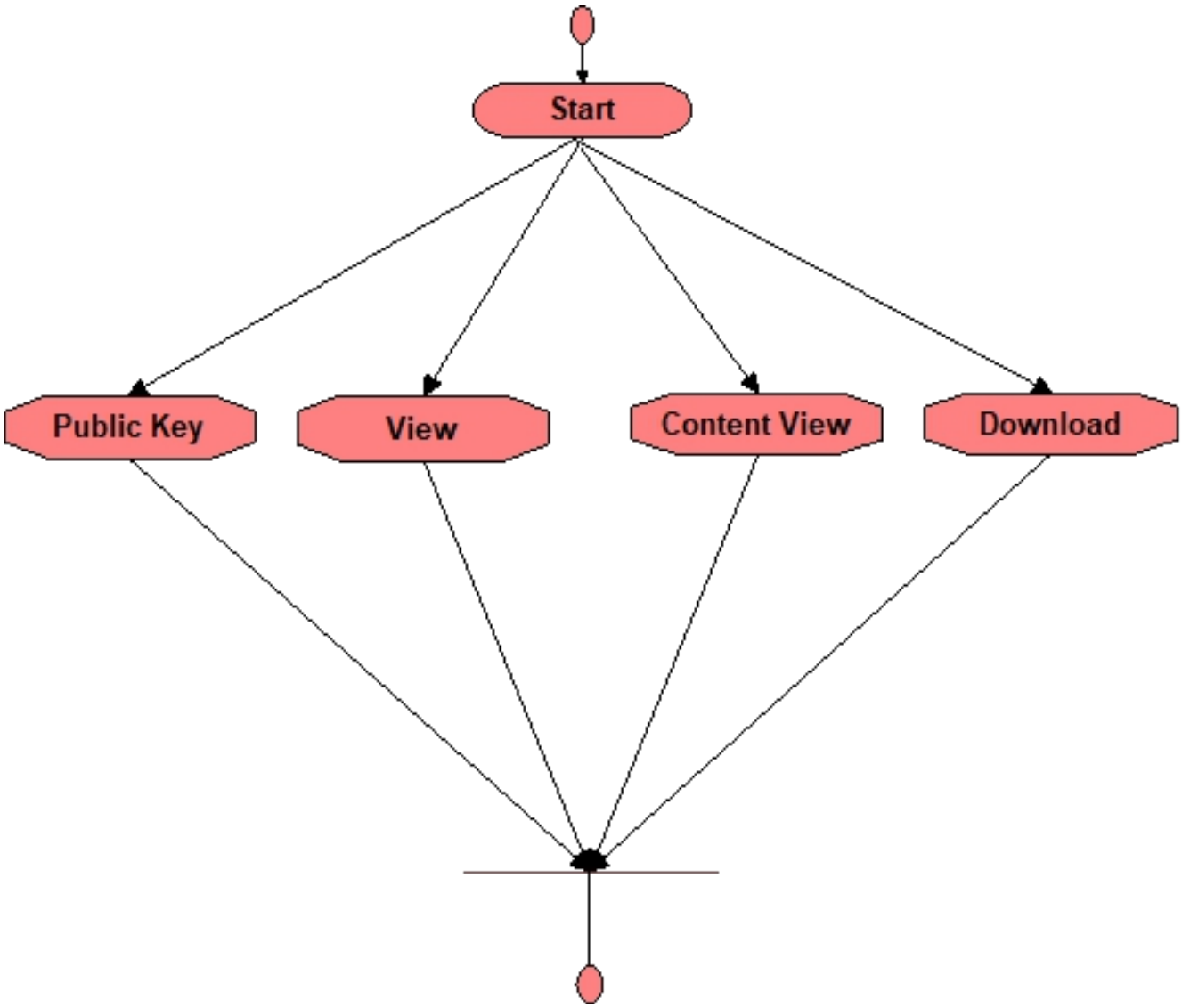


Fig 5.11 Activity Diagram Auditor

5.3.3 Data Flow Diagram User

Data drive business activities. They can trigger events and processes to provide information useful to personnel. System analysis recognizes the central role of business data in the organization. The following

shows the flow of data through business processes, which is the purpose of data flow analysis, tells us a great deal about how organization objectives are accomplished.

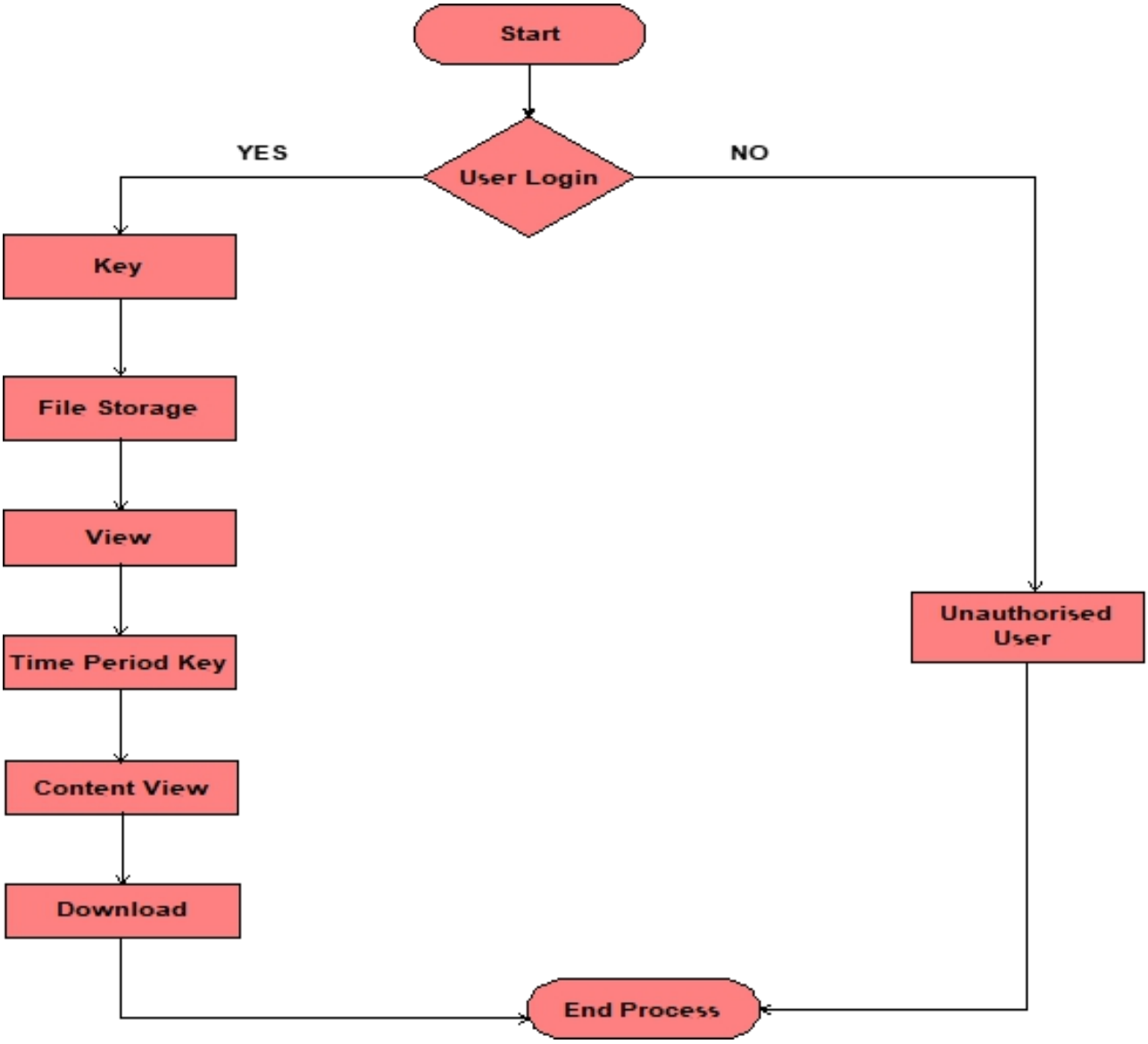


Fig 5.12 Data Flow Diagram User

5.3.4 Data Flow Diagram Auditor

Data drive business activities. They can trigger events and processes to provide information useful to personnel. System analysis recognizes the central role of business data in the organization. The following

shows the flow of data through business processes, which is the purpose of data flow analysis, tells us a great deal about how organization objectives are accomplished.

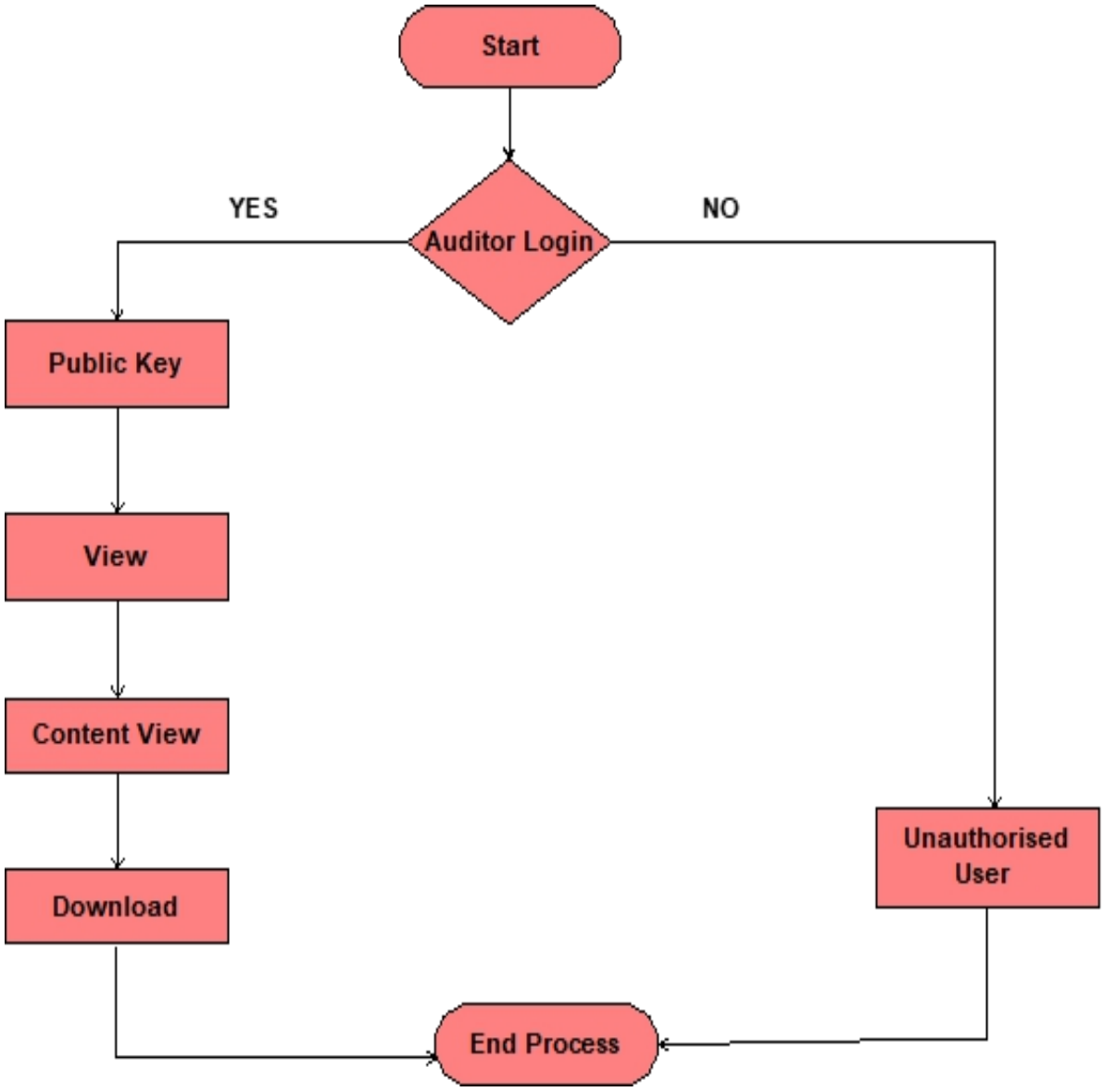


Fig 5.13 Data Flow Diagram Auditor

5.3.5 Component Diagram User

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 are often used to model the static implementation view of a system**.**

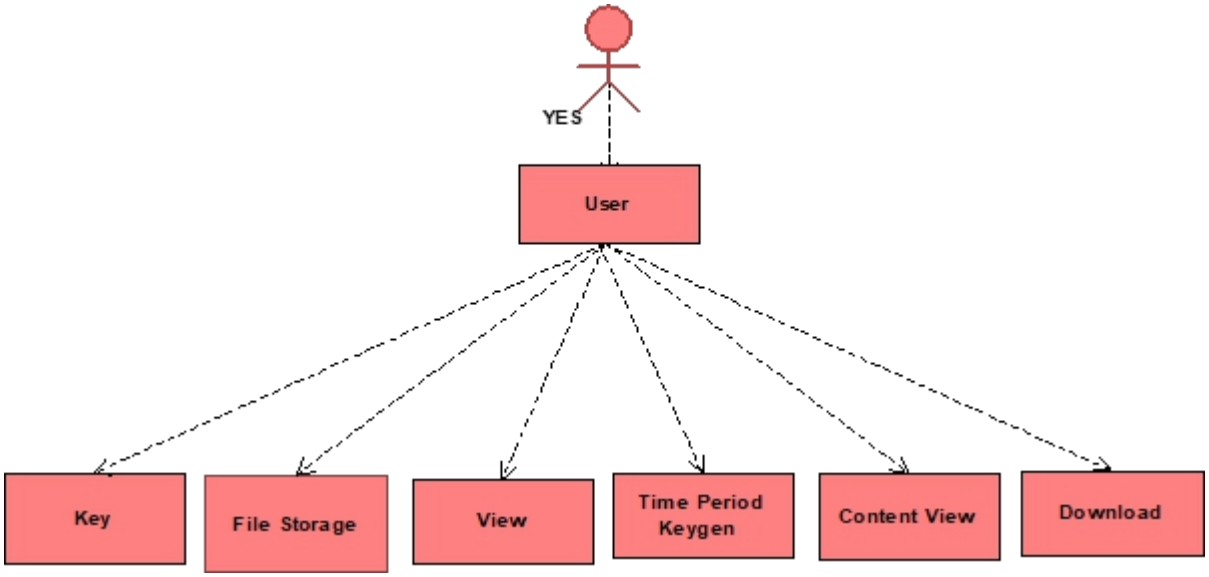


Fig 5.14. Component Diagram User

5.3.6 Component Diagram Auditor

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 are often used to model the static implementation view of a system**.**

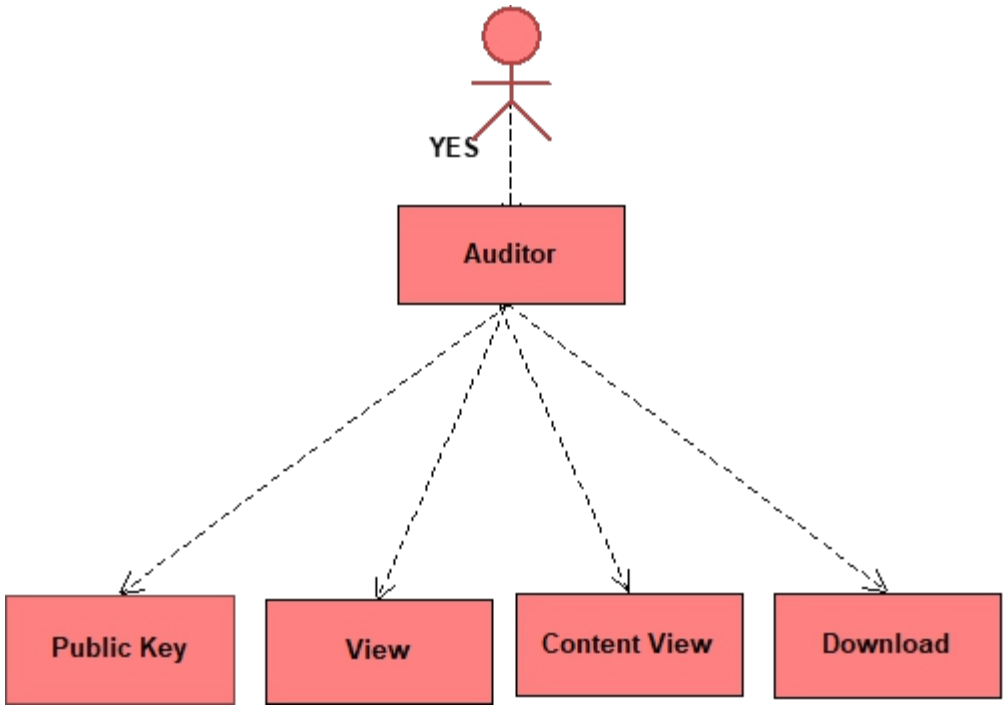


Fig 5.15 Component Diagram Auditor

5.3.7 Use Case Diagram User

A Use case diagram shows a set of use cases and actors and their relationships. Use case diagrams address the static Use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system. Use case diagram consists of Use case, actors, and the relationships between them.

- USE CASE: Use case is a description of a set of sequence of actions that a system performs that yields an observable result of value to a particular actor. A use case is used to structure the behavioral things in a model. A use case is realized by collaboration.
- ACTOR: Actor is the user of the system, who performs action on the system and to whom the system yields an observable result of value.

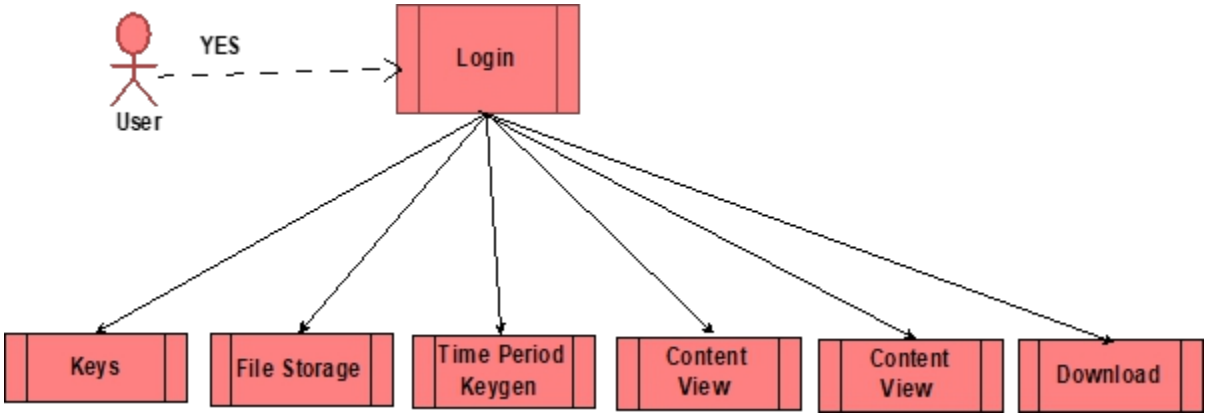


Fig 5.16 Use Case Diagram User

5.3.8 Use Case Diagram Auditor

A Use case diagram shows a set of use cases and actors and their relationships. Use case diagrams address the static Use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system. Use case diagram consists of Use case, actors, and the relationships between them.

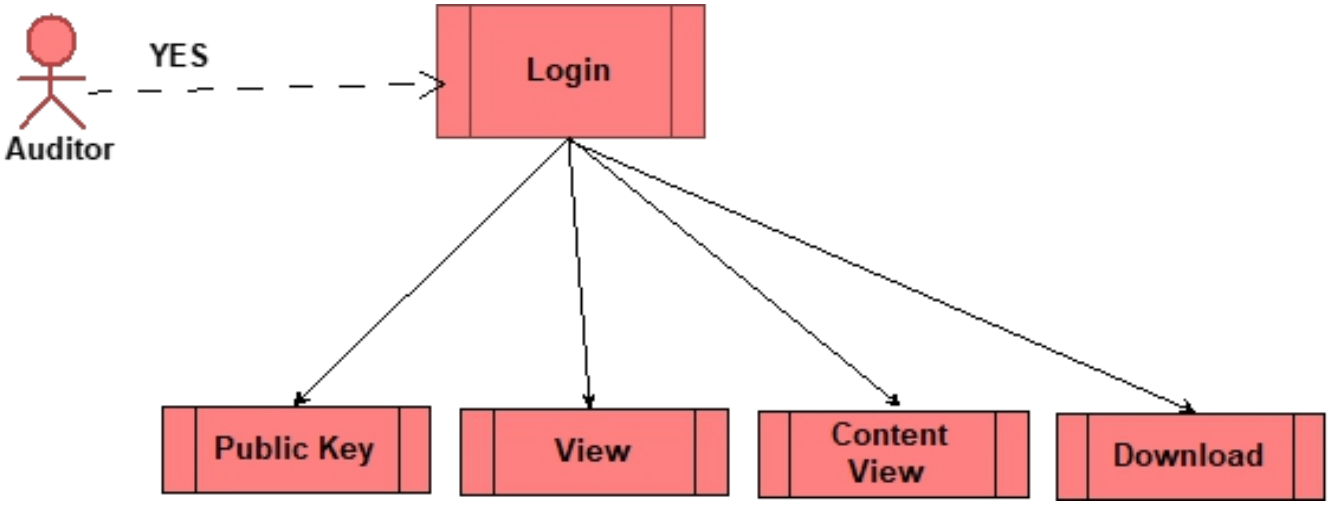


Fig 5.17 Use Case Diagram Auditor

5.3.9 Sequence Diagram User

The sequence diagram is an interaction diagram that emphasizes the time ordering of messages for modeling a real time system. Graphically, a sequence diagram is a table that shows objects arranged along the X axis and messages, ordered in increasing time, along the Y axis. Sequence Diagram consists of objects, links, lifeline, focus of control, and messages.

- OBJECT: Objects are typically named or anonymous instances of class. But may also represent instances of other things such as components, collaboration and nodes.
- LINK: A link is a semantic connection among objects i.e., an object of an association is called a link.
- LIFELINE: A lifeline is a vertical dashed line that represents the lifetime of an object.
- FOCUS OF CONTROL: A Focus of Control is a tall, thin rectangle that shows the period of time during which an object is performing an action.
- MESSAGES: A message is a specification of a communication between objects that conveys the information with the expectation that the activity will ensue.

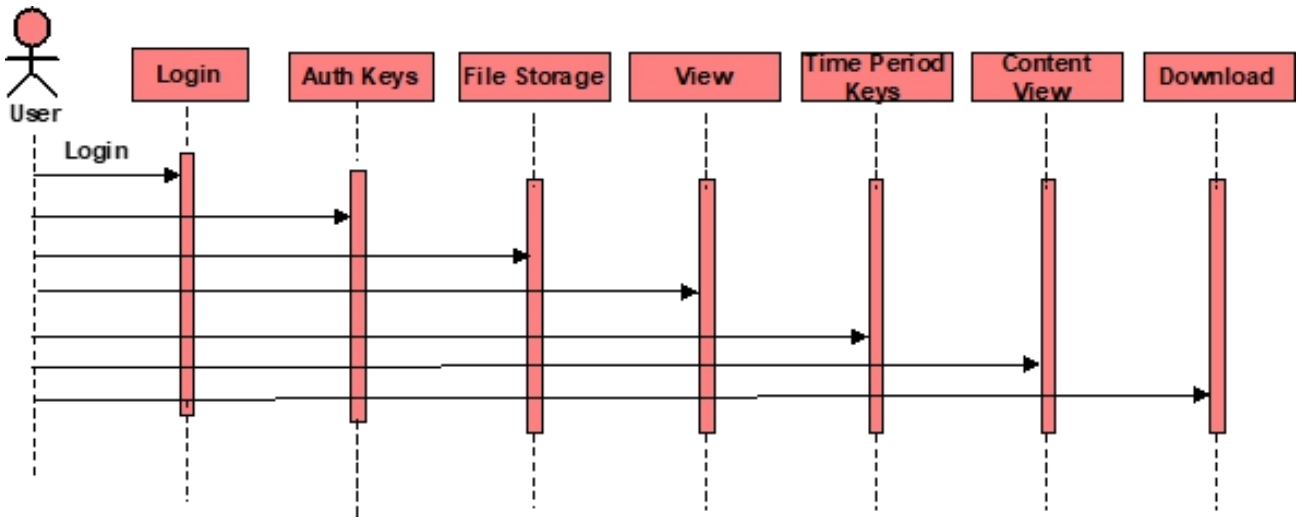


Fig 5.18 Sequence Diagram User

5.3.10. Sequence Diagram Auditor

The sequence diagram is an interaction diagram that emphasizes the time ordering of messages for modeling a real time system. Graphically, a sequence diagram is a table that shows objects arranged along

the X axis and messages, ordered in increasing time, along the Y axis. Sequence Diagram consists of objects, links, lifeline, focus of control, and messages.

- OBJECT: Objects are typically named or anonymous instances of class. But may also represent instances of other things such as components, collaboration and nodes.
- LINK: A link is a semantic connection among objects i.e., an object of an association is called a link.

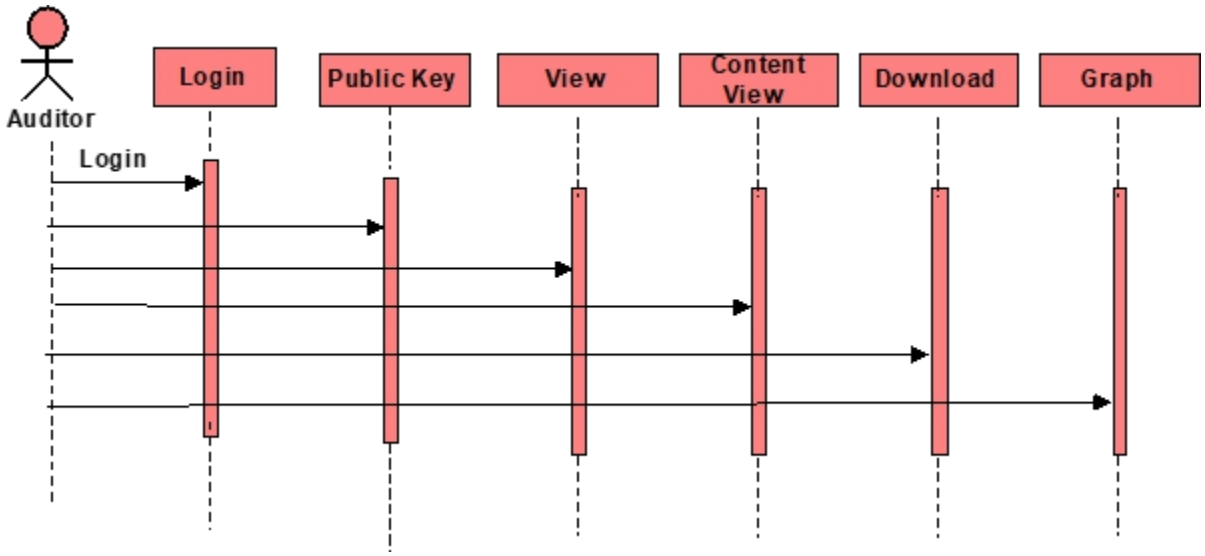


Fig 5.19 Sequence Diagram Auditor

6.OVERVIEW OF TECHNOLOGIES

6.1. 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 bytecode instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works. interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java bytecode instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works. 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.

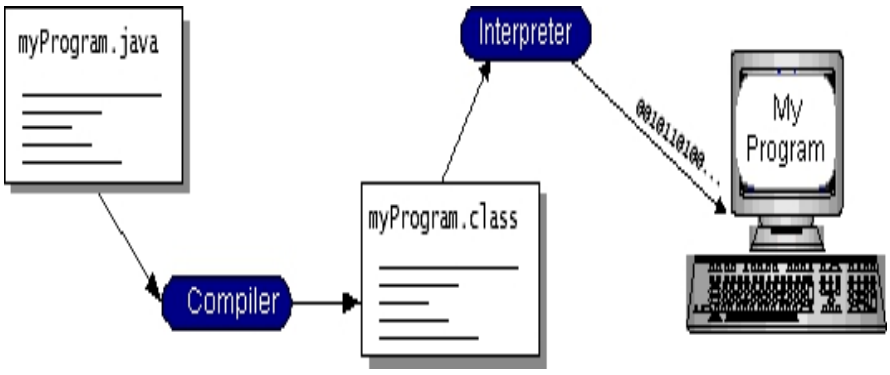


Fig.6.1: Working of Java Program

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

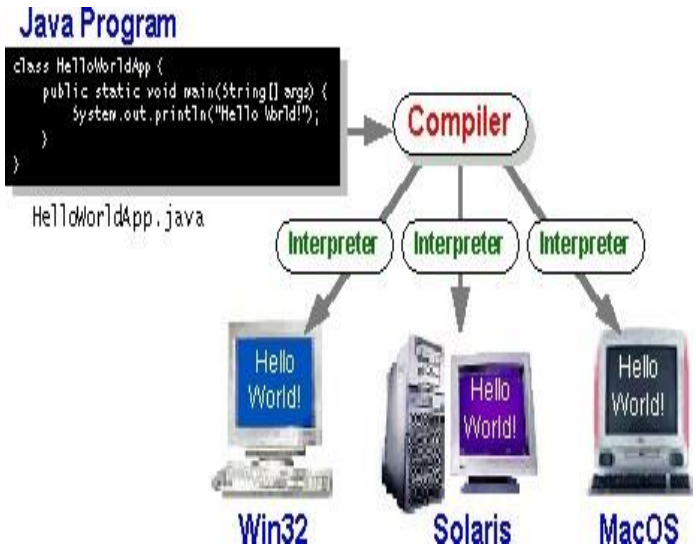


Fig.6.2: Implementation of Java Virtual Machine

6.2. 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. In the next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

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

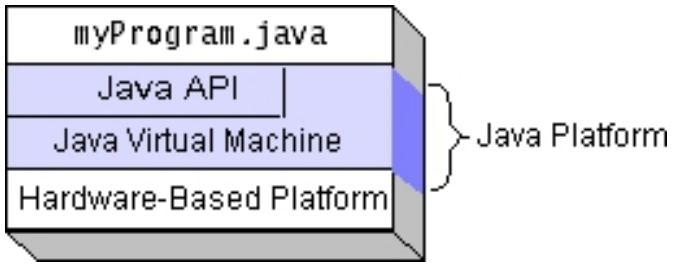


Fig.6.3: Program Running on the Java Platform

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 bytecode 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-enabledbrowser.

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-enabledbrowser.

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 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 provide 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 JavaBeans, can plug into existing component architectures.
- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC):** Provides uniform access to a wide range of relational databases.

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

6.3. 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 the 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 set up a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in the 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.

6.4. 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 design goals for JDBC are as follows

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 future tool vendors to generate JDBC code and to hide many of JDBC‘s complexities from the end user.

SQL Conformance:

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying

database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

JDBC must be implemental on top of common database interfaces:

The JDBC SQL API must ―sit‖ on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

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.

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.

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.

- **Map Visualizations**

  Charts showing values that relate to geographical areas. Some examples include:

  1. population density in each state of the United States,
  2. income per capita for each country in Europe,
  3. 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

- **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.

- **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.

- **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.

Tomcat 6.0 web server

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

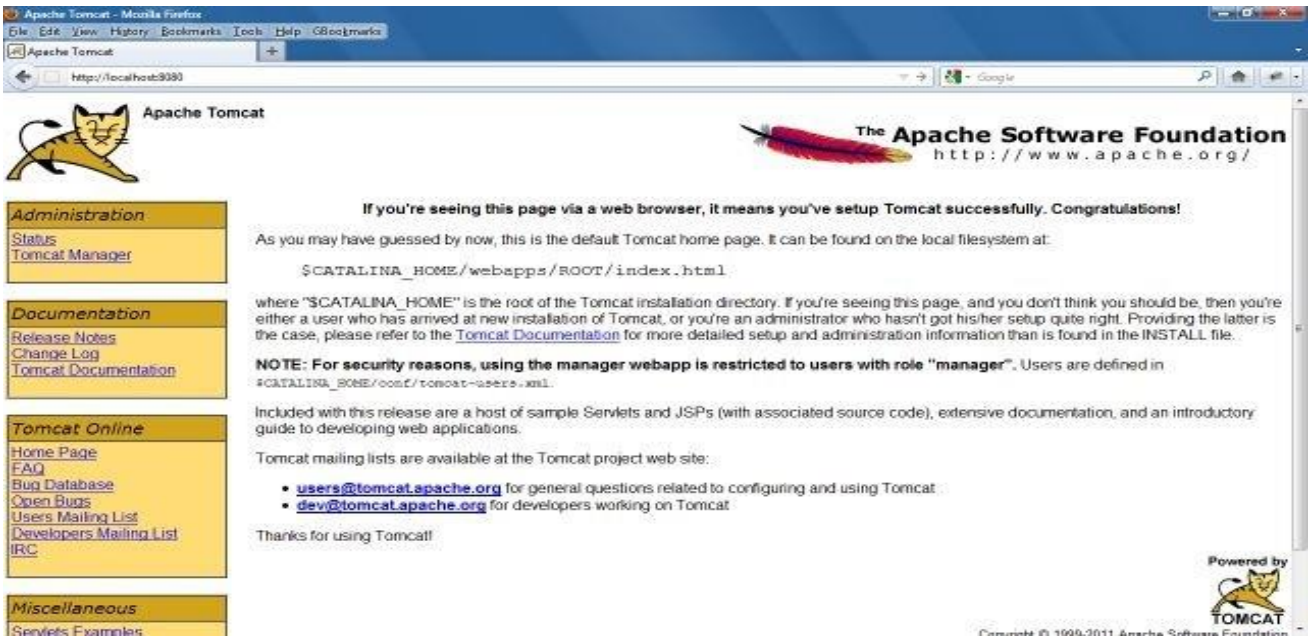


Fig Tomcat Web Server

7. OVERVIEW OF DBMS

7.1. DATA ABSTRACTION

Abstraction means to provide necessary information without considering the background details. There are three levels of abstraction for a DBMS.

- **Physical level:** It is the lowest level of abstraction, which describes how the data was actually stored on secondary devices such as disks and tapes.
- **Logical level**: It is a second level of abstraction, which describes what data are stored in the database, and what relationships exist among those data. Database Administrators decide what data is to be kept in the database.
- **View level:** It is the highest level of abstraction, which describes only a part of the entire database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.

7.2 INSTANCES AND SCHEMAS

The collection of information stored in a database at a particular moment is called an instance. The overall design of a database is called a schema.

7.3. DATA MODELS

A Data Model is a collection of conceptual tools for describing data, data relationship, data semantic and consistency constraints. Various data models available are discussed below.

7.3.1. The Entity Relationship Model

E-R model is a data model used to describe the data involved in a real world enterprise. It describes the data in the form of entities and relationships. An entity is a ‘thing’ (or ‘object’) in the real world that can be easily distinguishable from other things. A relationship is an association among several entities.

7.3.2. Relational Model

The Relational Model uses a collection of tables to represent both data and the relationships among the data. Each table has multiple columns, and each column has a unique name.

7.4. DATABASE LANGUAGES

A database system provides data definition language and data manipulation language.

7.4.1. Data Definition Language

Data Definition Language (DDL) consists of a set of definitions used to specify database schema. Execution of DDL statements results in a set of tables. These tables are stored in a specific area known as a data dictionary or data directory. A data directory contains Metadata. Metadata is data about data.

7.4.2. Data Manipulation Language

Data manipulation is

- Retrieval of information stored in the database
- Insertion of new information into the database.
- Deletion of information from the database.
- Modification of information stored in the database.

Data Manipulation Language (DML) is a language that enables users to access or manipulate data. There are basically two types

- Procedural DMLs require a user to specify what data is needed and how to get that data.
- Declarative DMLs require the user to specify what data needed without specifying how to get those data.

7.5. MYSQL

7.5.1. Introduction

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

- **MYSQL is portable**

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

- **MYSQL is compatible**

  MYSQL commands can be used for communicating with IBM DB2 mainframe RDBMS that is different from MYSQL, that is MYSQL compatible with DB2. MYSQL RDBMS is a high performance fault tolerant DBMS, which is specially designed for online transaction processing and for handling large database applications.

- **Multithreaded server architecture**

  MYSQL adaptable multithreaded server architecture delivers scalable high performance for a very large number of users on all hardware architecture including symmetric multiprocessors (sumps) and loosely coupled multiprocessors. Performance is achieved by eliminating CPU, I/O, memory and operating system bottlenecks and by optimizing the Sql Server 2005, DBMS server code to eliminate all internal bottlenecks.

7.5.3. Features of MYSQL

Most popular RDBMS in the market because of its ease of use

- Client/server architecture.
- Ensuring data integrity and data security.
- Parallel processing support for speed up data entry and online transaction processing used for applications.

Dr.E.F.CODD's RULES

These rules are used for valuing a product to be called relational database management systems. Out of 12 rules, a RDBMS product should satisfy at least 8 rules, +rule called rule 0 that must be satisfied.

RULE 0. FOUNDATION RULE

All information in relational databases is represented at logical level in only one way as values in tables.

RULE 2. GUARANTEED ACCESS

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

RULE 3. SYSTEMATIC TREATMENT OF NULL VALUES

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

RULE 4. DYNAMIC ONLINE CATALOG BASED RELATION MODEL

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

RULE 5. COMPREHENSIVE DATA SUB LANGUAGE

A relational system may support several languages and various models of terminal use. However there must be one language whose statement can express all of the following Data Definitions, View Definitions, Data Manipulations, Integrity, Constraints, Authorization and transaction boundaries.

RULE 6. VIEW UPDATING

Any view that is theoretical can be updatable if changes can be made to the tables that affect the desired changes in the view.

RULE 7. HIGH LEVEL UPDATE, INSERT and DELETE

The capability of handling a base relational or derived relational as a single operand applies not only retrieval of data also to its insertion, updating, and deletion.

RULE 8. PHYSICAL DATA INDEPENDENCE

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

RULE 9. LOGICAL DATA INDEPENDENCE

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

RULE 10. INTEGRITY INDEPENDENCE

Integrity constraints specific to a particular database must be definable in the relational data stored in the catalog, not in the application program.

RULE 11. DISTRIBUTED INDEPENDENCE

Whether or not a system supports database distribution, it must have a data sublanguage that can support distributed databases without changing the application program.

RULE 12. NON SUBVERSION

If a relational system has a low level language, that low language cannot use subversion or bypass the integrity rules and constraints expressed in the higher level relational language.

MYSQL SUPPORTS THE FOLLOWING CODD'S RULES:

- Rule 1: Information Rule (Representation of information)-YES.

- Rule 2: Guaranteed Access-YES.
- Rule 3: Systematic treatment of Null values-YES.
- Rule 4: Dynamic on-line catalog-based Relational Model-YES.
- Rule 5: Comprehensive data sublanguage-YES.
- Rule 6: View Updating-PARTIAL.
- Rule 7: High-level Update, Insert and Delete-YES.
- Rule 8: Physical data Independence-PARTIAL.
- Rule 9: Logical data Independence-PARTIAL.
- Rule 10: Integrity Independence-PARTIAL.
- Rule 11: Distributed Independence-YES.
- Rule 12: Non-subversion-YES.

8. IMPLEMENTATION

In the implementation phase software development is concerned with translating design specifications into source code. The primary goal of implementation is to write the source code internal documentation so that conformance of the code to its specification can be easily verified, and so that debugging, testing and modifications are erased. This goal is achieved by making the source code as clear and straightforward as possible. Simplicity, clarity and elegance are the hallmarks of good programs. Obscurity, cleverness and complexity are indications of inadequate design and misdirected thinking.

Source code clarity is enhanced by strutted techniques, by good coding style, by appropriate documents, by internal comments, and by the features provided in the modern programming languages.

The main aim of structured coding is to adhere to single entry, single exit constructs in the majority of situations since it allows one to understand program behavior by reading the code from beginning to end. But

strict adherence to this construct may cause problems and it raises concerns for the time and space efficiency of the code. In some cases, single entry and single exit programs will require repeated code segments or repeated subroutines calls. In such cases, the usage of this construct would prevent premature loop exits and branching to exception handling code. So, in certain situations we violate this construct to acknowledge the realities of implementation although our intent is not encouraging poor coding style.

In computer programming, coding style is manifested in the patterns used by programmers to express a desired action or outcome. Good coding style can overcome the deficiencies of primitive programming languages, while poor style can defeat the intent of an excellent language. The goal of good coding style is to provide easily understood straightforward, elegant code.

Every good coding style performs the following Do‘s

- Introduce user-defined data types to model entities in the problem domain.
- Use a few standards, agreed-upon control statements.
- Hide data structures behind access functions.
- Use goto's in a disciplined way
- Isolate machine dependencies in a few routines.
- Use indentation, parenthesis, blank lines and borders around comment blocks to enhance readability.
- Carefully examine the routines having fewer than 5 or more than 25 executable statements.

The following are the Don‘ts of good coding style

- Avoid null statements.
- Don‘t put nested loops very deeply.
- Carefully examine routines having more than five parameters.
- Don‘t use an identifier for multiple purposes.

Adherence implementation standards and guidelines by all programmers on a project results in a product of uniform quality. Standards were defined as those that can be checked by an automated tool. While determining adherence to a guideline requires human interpretation. A programming standard might specify items such as:

- The nested depth of the program constructs will not execute five levels.
- The go to statements will not be used.
- Subroutine lengths will not exceed 30 Lines.

Implementation was performed with the following objectives.

- Minimize the memory required.
- Maximize output readability or clarity.
- Maximize source text readability.
- Minimize the number of source statements.
- To ease modification of the program.
- To facilitate formal verification of the program.
- To put the tested system into operation while holding costs, risks and user irritation to minimum.

Supporting documents for the implementation phase include all base-lined work products of the analysis and design phase.

9. SOURCE CODE

Fileupload.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
 <html>
<head>
<title>File Upload</title>
<link href="css/style.css" rel="stylesheet" type="text/css">
</head>
<body>

	<div id="header">
		<div>
		<p class="sp"> Enabling Cloud Storage Auditing with <br/>Key-Exposure Resistance </p>
<ul class="navigation">
<!-- <li><a
	href="/CloudStorageAuditKey/index.html">Home</a></li> -->
	<li><a class="active" href="fileupload.jsp">File Storage</a></li>
	<li><a href="fileview.jsp">File View</a></li>
	<!-- <li><a href="timesecretkey.jsp">Timesecret Key</a></li> -->
<li><a href="logout.jsp">Logout</a></li>
</ul> </div></div> <div id="body"> <div class="content">
	<div> <div> <p>
<form action="upload1" enctype="multipart/form-data" name="form" method="post"
onSubmit="return validation()">
```

```
<table width="70%" align="center" cellpadding="5px" cellspacing="5px">
<tr> <th colspan="2" align="center"> <p>
<strong><font color="#D1102D" size="6">File Storage</font></strong> </p> </th> </tr>
 <tr>
        <td width="35%"> <p>
<strong><font color="#333399" size="4">Public Key </p>
        </font></strong> </p> </td> <td><p>
<input type="text" name="pkey" required> </p></td> </tr>
 <tr>
<td align="center"><p><strong><font color="#333399" size="4">File Name</font></strong></p></td>
<td><input type="text" name="fname" required></td> </tr> <tr>
<td align="center"><p><strong><font color="#333399" size="4">File </font></strong></p></td>
<td><input type="file" name="fcont" required></td> </tr>
<tr>
<td height="36" align="right"><input type="reset" value="Reset" /></td>
<td align="left"><input type="submit" value="Submit" /></td> </tr>
</table> </form> </p> </div> </div> </div>
<div class="sidebar" align="left">
<table>
        <tr style="align:left">
<td><font size="6" color="#FF0040"> <strong>Cloud Sharing</strong></font></td> </tr>
        <tr style="align:left">
<td><ul><a href="https://www.cloudme.com/en"><font size="5" color="#0000FF"><strong>
```

```
        <li type="square">Cloud Me</li></strong></font></a></ul></td> </tr>
</table>
<img src="images/fstore.png" >
</div> </div>
<div id="footer">
<div class="abc" align="center">
<p>&copy; Copyright 2015. All Rights Reserved</p>
 </div> </div>
</body>
</html>
```

Graph.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@ page language="java" import="java.sql.*,com.dbasecon.Database On" errorPage="" %>
<%@ page language="java" import="java.io.File"%>
<%@ page language="java" import="org.jfree.chart.ChartFactory"%>
<%@ page language="java" import="org.jfree.chart.ChartUtilities"%>
<%@ page language="java" import="org.jfree.chart.JFreeChart"%>
<%@ page language="java" import="org.jfree.chart.plot.PlotOrientation"%>
<%@ page language="java" import="org.jfree.data.jdbc.JDBCCategoryDataset"%>

 <html>
<head>
        <title>File Content</title>
        <link href="css/style.css" rel="stylesheet" type="text/css">
```

```
        <style>
        </style>
</head>
<body>
<%
int user=0,files=0,keygen=0,keynote gen=0;

Connection con=null;
// Number of Users
try
{
        con=Databasecon.getConnection();
        String query="select Uname from registration";
        PreparedStatement pstmt=con.prepareStatement(query);
        ResultSet rs=pstmt.executeQuery();
        while(rs.next())
        {
                user++;
        }
}
catch(Exception e)
{
        e.printStackTrace();
}

System.out.println("No of users : "+user);

// Number of Files
try
{
        con=Databasecon.getConnection();
```

```
        String query="select fname from file";
        PreparedStatement pstmt=con.prepareStatement(query);
        ResultSet rs=pstmt.executeQuery();
        while(rs.next())
        {
                files++;
        }
}
catch(Exception e)
{
        e.printStackTrace();
}
System.out.println("No of Files : "+files);

//KeyGen
try
{
        con=Databasecon.getConnection();
        String query="select status from file where status='Generated'";
        PreparedStatement pstmt=con.prepareStatement(query);
        ResultSet rs=pstmt.executeQuery();
        while(rs.next())
        {
                keygen;
        }
}
catch(Exception e)
{
        e.printStackTrace();
}
System.out.println("KeyGen : "+keygen);
```

```
//Key Not Gen
try
{
	con=Databasecon.getConnection();
	String query="select status from file where status='Key Not Gen'";
	PreparedStatement pstmt=con.prepareStatement(query);
	ResultSet rs=pstmt.executeQuery();
	while(rs.next())
	{
		keynote n++;
	}
}
catch(Exception e)
{
	e.printStackTrace();
}
System.out.println("Key Not Gen : "+keynote gen);


//Graph Table Update
//user=0,files=0,keygen=0,keynote gen=0;
try
{
	con=Databasecon.getConnection();
	String query="update graph set Users='"+user+"' , Files='"+files+"',Keygen='"+keygen+"',Keynote
Gen='"+keynote gen+"' where sid='1'";
	PreparedStatement pstmt=con.prepareStatement(query);
	int x=pstmt.executeUpdate();
	if(x>0)
	{
		System.out.println("Graph Updated");
```

```
        }else
        {
                System.out.println("Graph Not Updated");
        }
}
catch(Exception e)
{
        e.printStackTrace();
}
//Graph Creation

//Graph
        String gph="Graph";
String dirname=getServletContext().getRealPath("");
        String dir=dirname"/"+gph;
System.out.println("Directory"+dir);
         File fold=new File(dir);
                                        if(!fold.exists())
                                        {
                                                fold.mkdir();
                                                System.out.println("Folder Created");

                                        }
                                        try
                                        {
                                         String query="select * from graph";
JDBCCategoryDataset dataset=newJDBCCategoryDataset("jdbc:mysql://localhost:3306/
cloud storage key","com.mysql.jdbc.Driver","root","admin");
        dataset.executeQuery(query);
JFreeChart chart = ChartFactory .createBarChart3D("Users and Files with Key Generation", "Graph Details",
"Numbers",dataset, PlotOrientation.VERTICAL, true, true, false);
```

```
ChartUtilities.saveChartAsJPEG(new File(dir+"/"+"chart2.jpg"), chart, 800, 400);
                                        }
                                        catch (Exception e)
                                        {
                                                e.printStackTrace();
                                        System.out.println("Problem in creating chart.");
                                        }
```

Filecontview.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
 <%@ page language="java" import="java.sql.*,com.dbasecon.Database On" errorPage="" %>
<%@ page language="java" import="java.io.*"%>
<%
String filename=request.getParameter("fname");
String pubkey=request.getParameter("pbkey");
String tskey1=request.getParameter("tskey");
System.out.println("Filename :" +filename);
System.out.println("Filename pub :" +pubkey);
```

```
System.out.println("Filename tskey :" +tskey1);

session.setAttribute("fname",filename);

session.setAttribute("pkey",pubkey);

session.setAttribute("tskey",tskey1);

response.sendRedirect("fcontview.jsp");

 %>
```

10. TESTING

10.1. SOFTWARE TESTING TECHNIQUES

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting anomaly for the software engineer.

10.1.1. Testing Objectives

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a probability of finding an as yet
3. undiscovered error.
4. A successful test is one that uncovers an undiscovered error.
5. These above objectives imply a dramatic change in view port.

Testing cannot show the absence of defects, it can only show that software errors are present.

10.1.2. Test Case Design

Any engineering product can be tested in one of two ways:

White Box Testing

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

Basis Path Testing

- Flow graph notation
- Cyclomatic Complexity

Deriving test cases Control Structure Testing

- Condition testing
- Data flow testing
- Loop testing

Black Box Testing

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

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis
- Comparison testing
- Graph matrices

10.2. SOFTWARE TESTING STRATEGIES

A Strategy for software testing integrates software test cases into a series of well planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers to the set of activities that ensure that the software that has been built is traceable to customer‘s requirements.

10.2.1. Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

10.2.2. Integration Testing

Integration testing is a systematic technique for constructing the program structure, while conducting tests to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

10.2.3. Functional Testing

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 identifying 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.

10.2.4. 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.

10.2.5. Security Testing

Attempts to verify the protection mechanisms built into the system

10.2.6. Performance Testing

This method is designed to test runtime performance of software within the context of an integrated system.

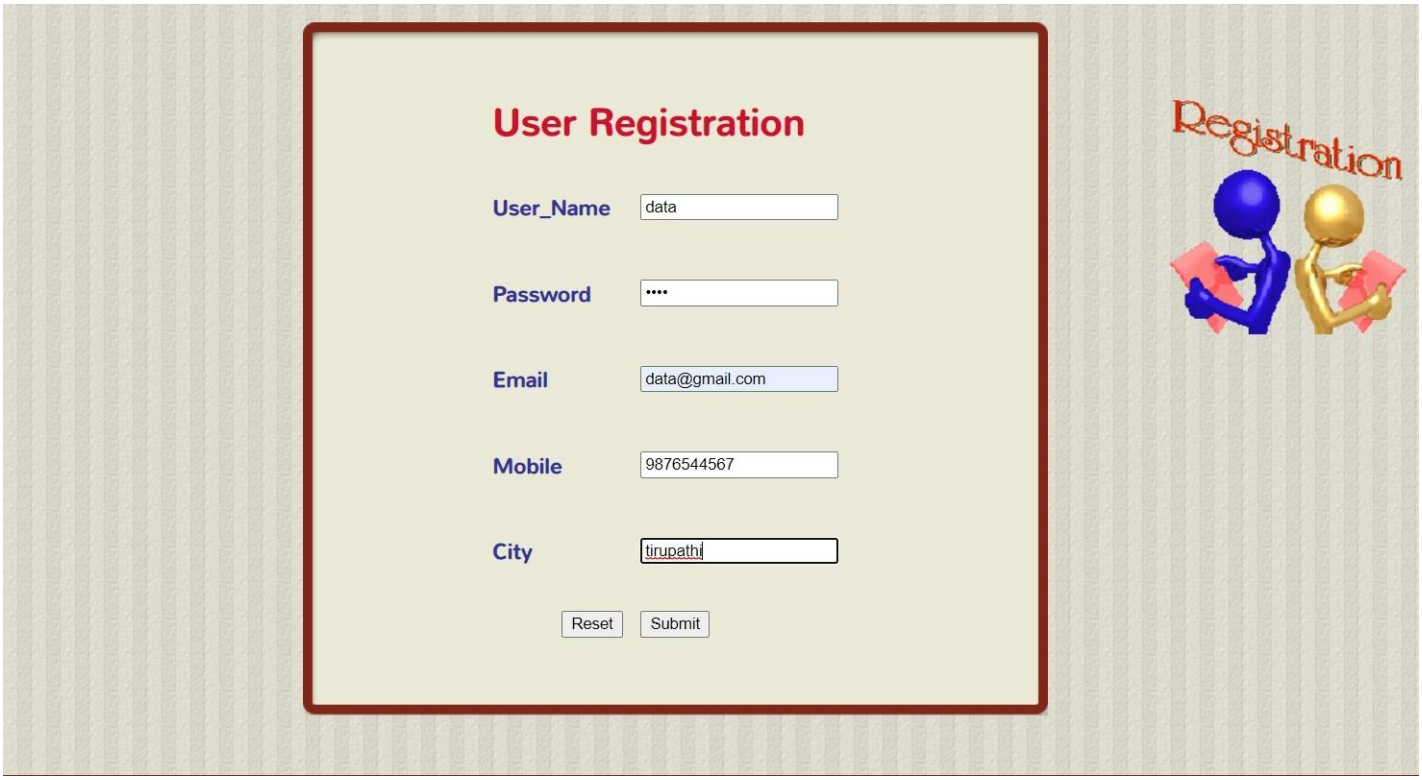
10.2.7 Acceptance Testing

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.

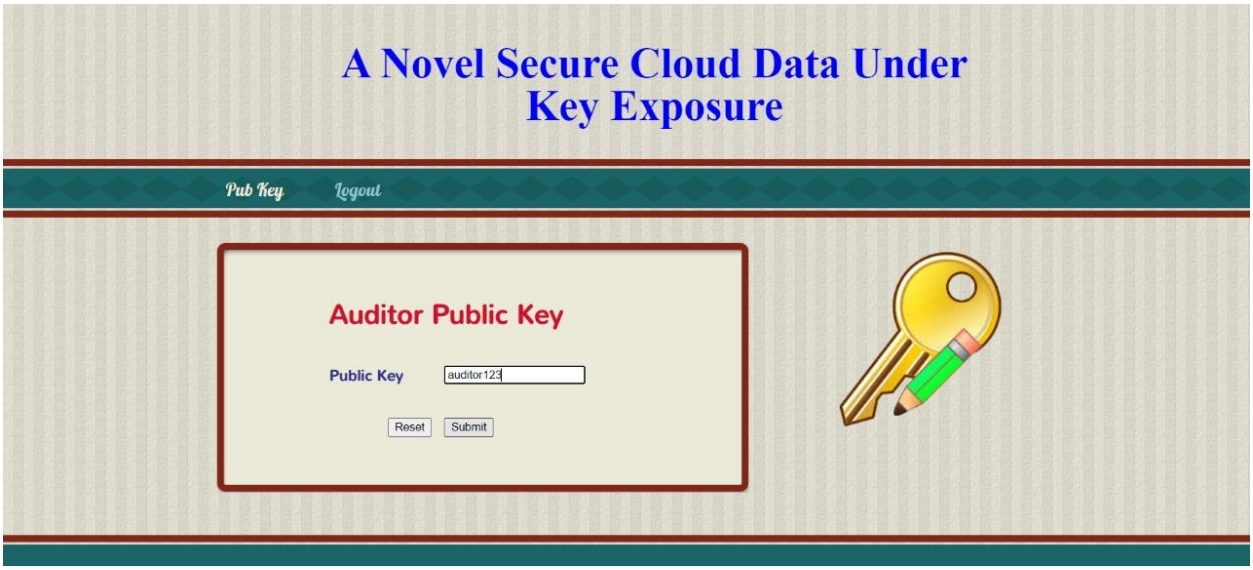
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

11. OUTPUT SCREENS

Screen 11.1. Home page User Login



Screen 11.2. User Registration

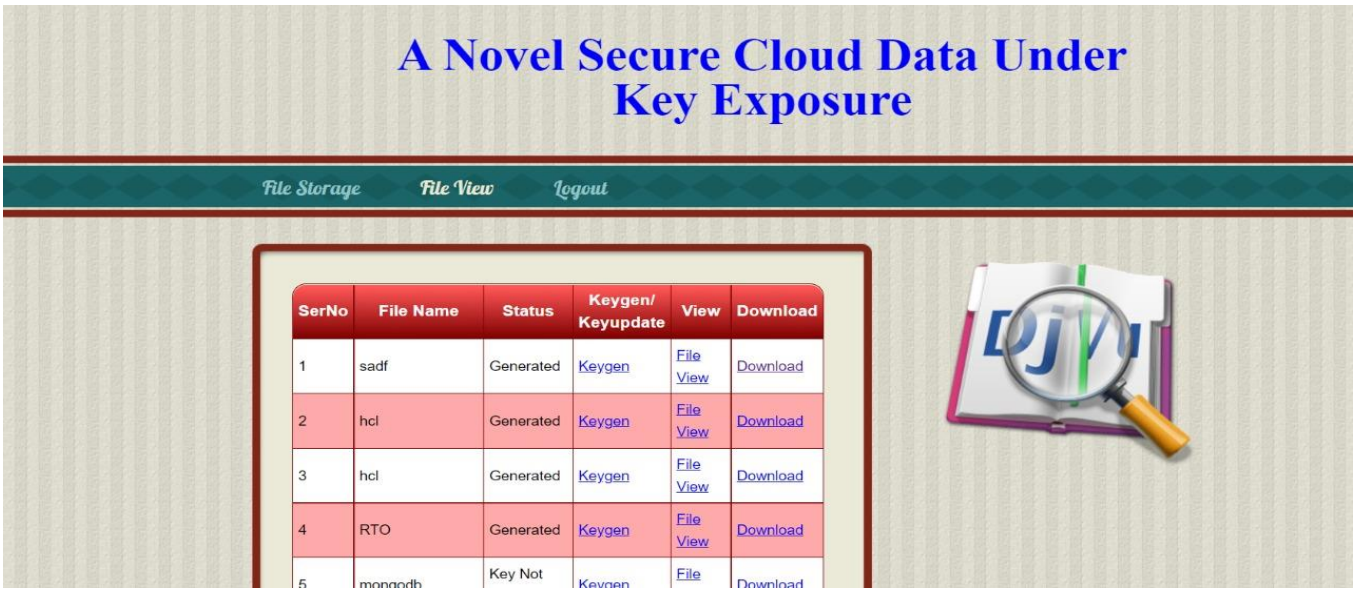


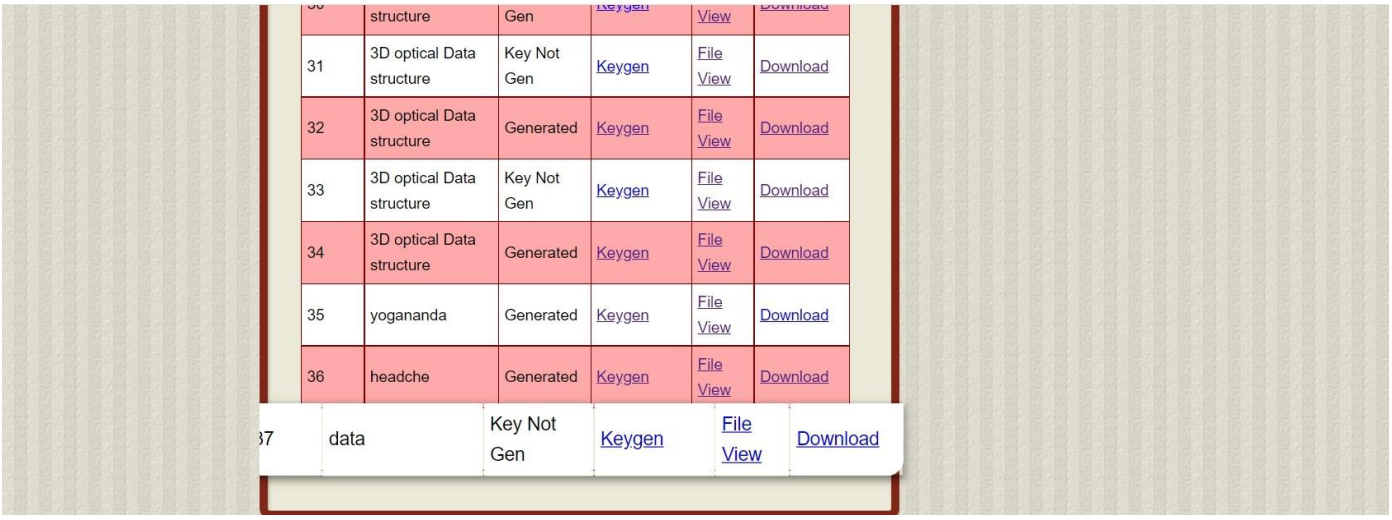
Screen 11.3. User Registered Successfully

Screen 11.4. User Login

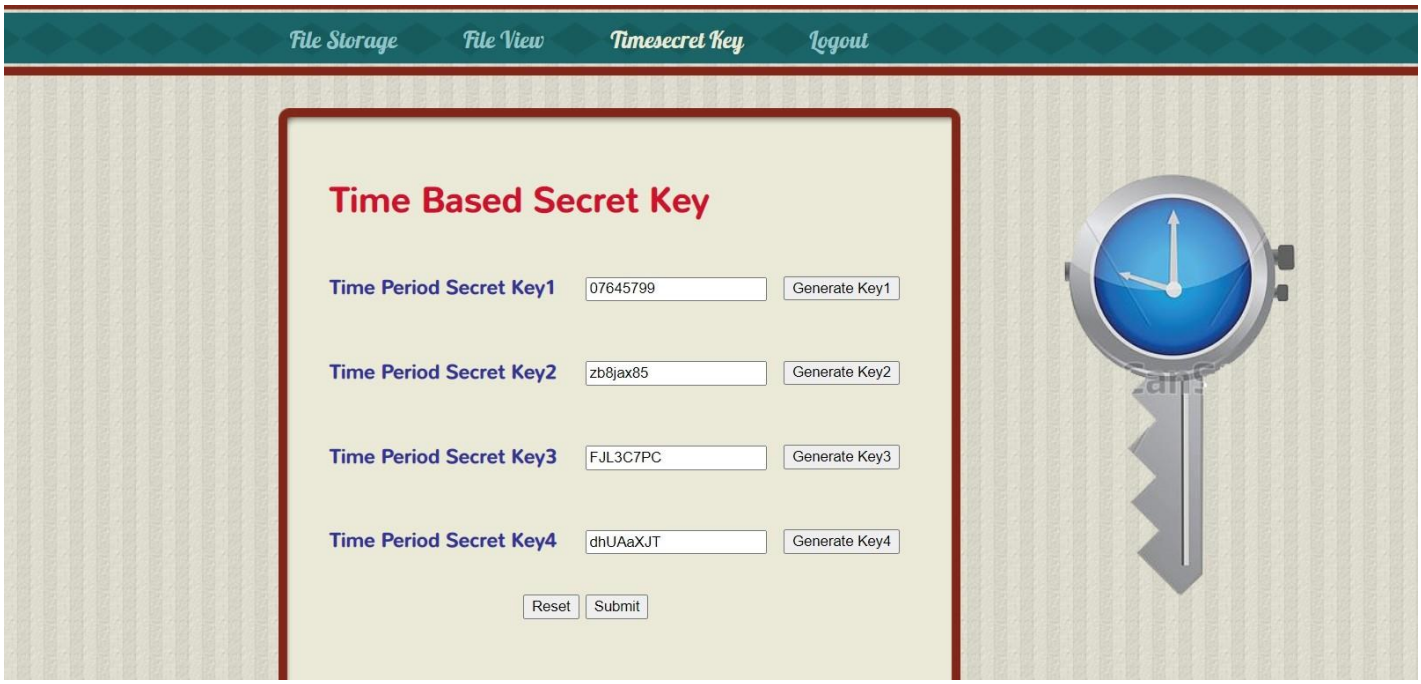
Screen 11.5. Authentication Keys

Screen 11.6. File Storage

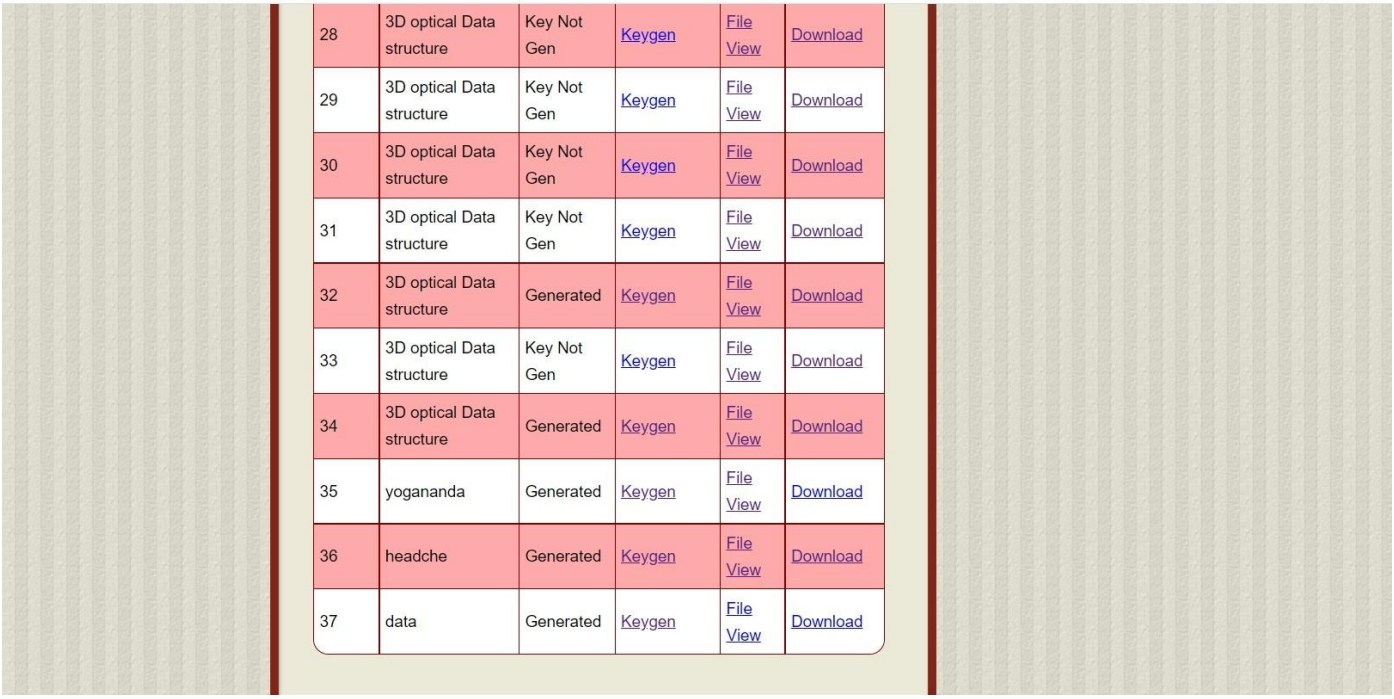




Screen 11.7. File View



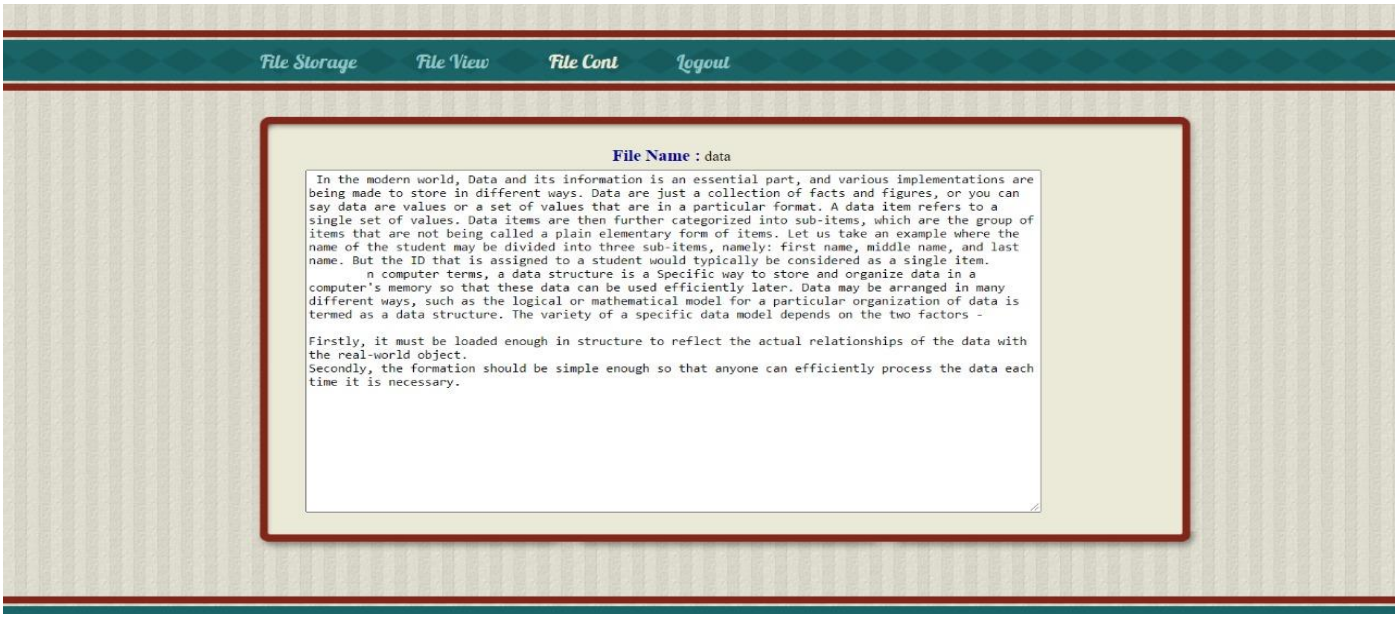
Screen 11.8. Time Based Secret Key



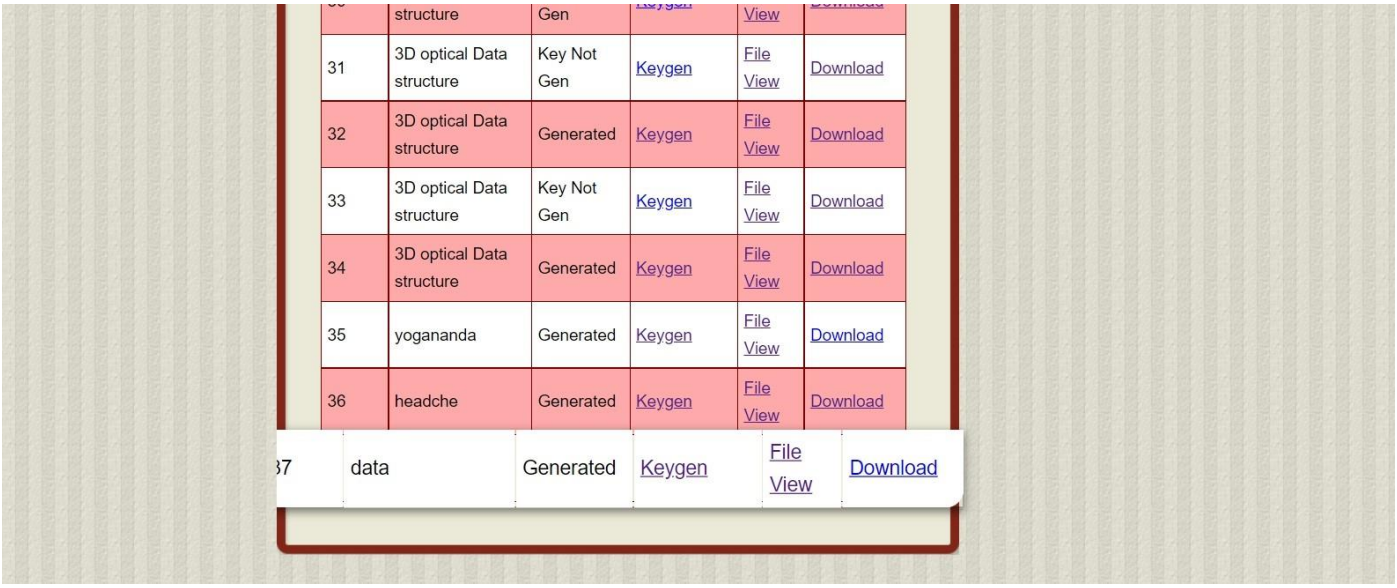
28	3D optical Data structure	Key Not Gen	Keygen	File View	Download
29	3D optical Data structure	Key Not Gen	Keygen	File View	Download
30	3D optical Data structure	Key Not Gen	Keygen	File View	Download
31	3D optical Data structure	Key Not Gen	Keygen	File View	Download
32	3D optical Data structure	Generated	Keygen	File View	Download
33	3D optical Data structure	Key Not Gen	Keygen	File View	Download
34	3D optical Data structure	Generated	Keygen	File View	Download
35	yogananda	Generated	Keygen	File View	Download
36	headche	Generated	Keygen	File View	Download
37	data	Generated	Keygen	File View	Download

Screen 11.9. Key Generated

Screen 11.10. File View Content



Screen 11.11. File View Data

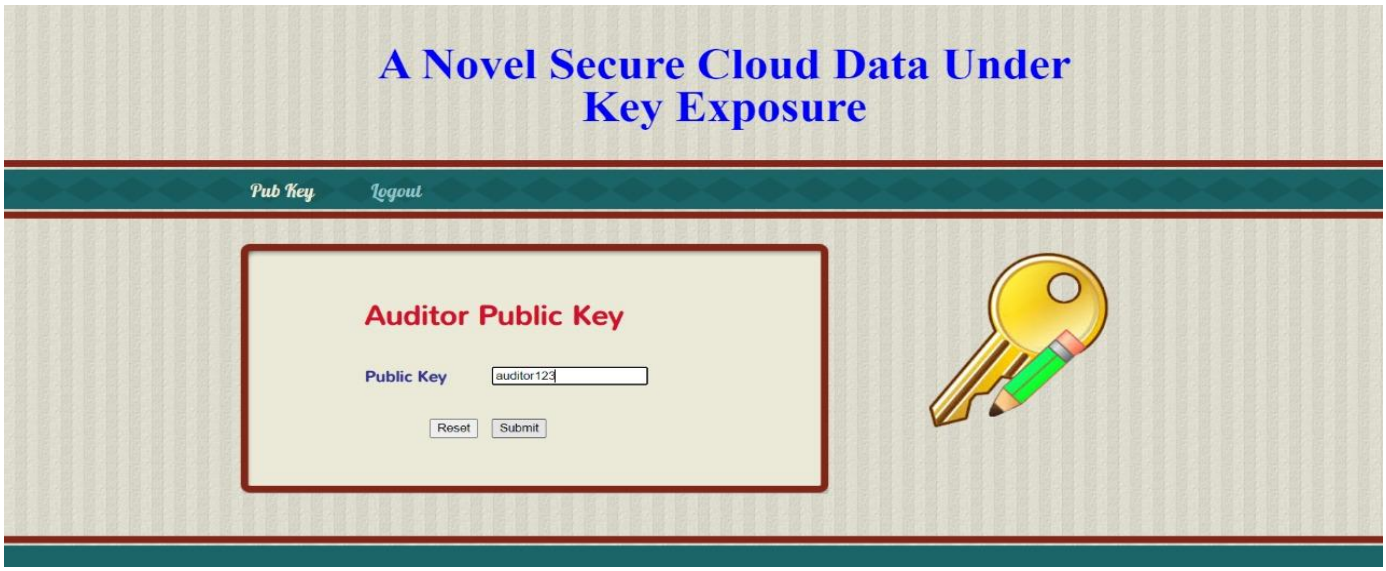


	structure	Gen		View	
31	3D optical Data structure	Key Not Gen	Keygen	File View	Download
32	3D optical Data structure	Generated	Keygen	File View	Download
33	3D optical Data structure	Key Not Gen	Keygen	File View	Download
34	3D optical Data structure	Generated	Keygen	File View	Download
35	yogananda	Generated	Keygen	File View	Download
36	headche	Generated	Keygen	File View	Download
37	data	Generated	Keygen	File View	Download

Screen 11.12. File Download

Screen 11.13. File Downloaded

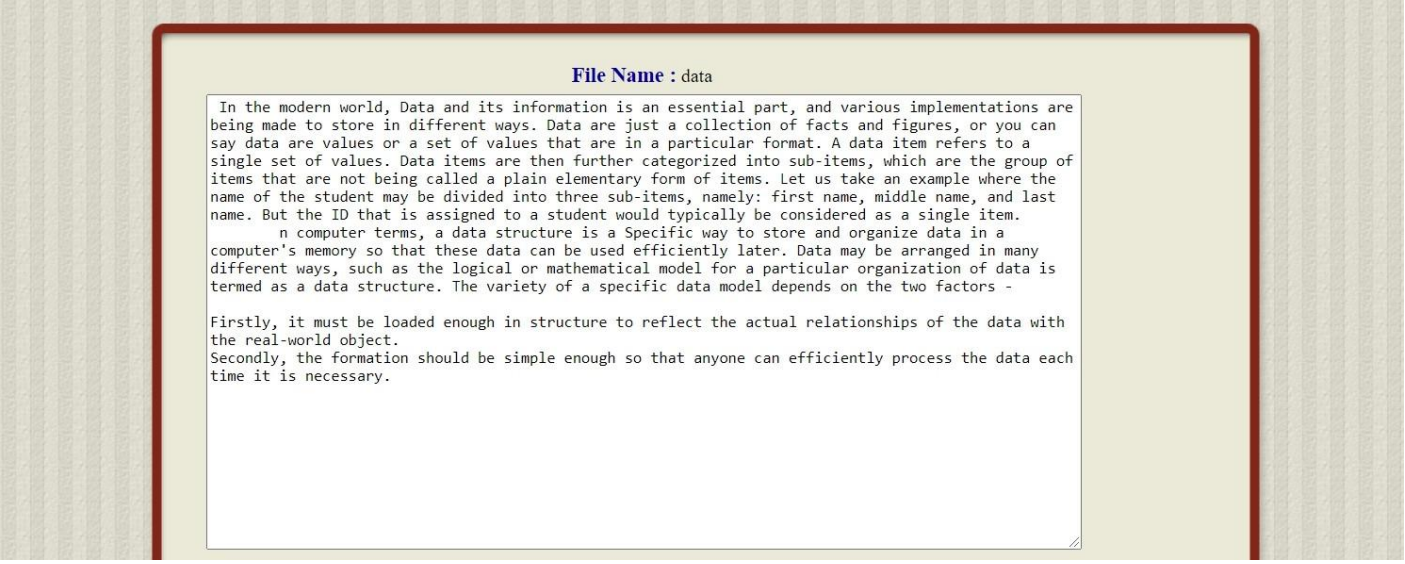
Screen 11.14. Auditor Login



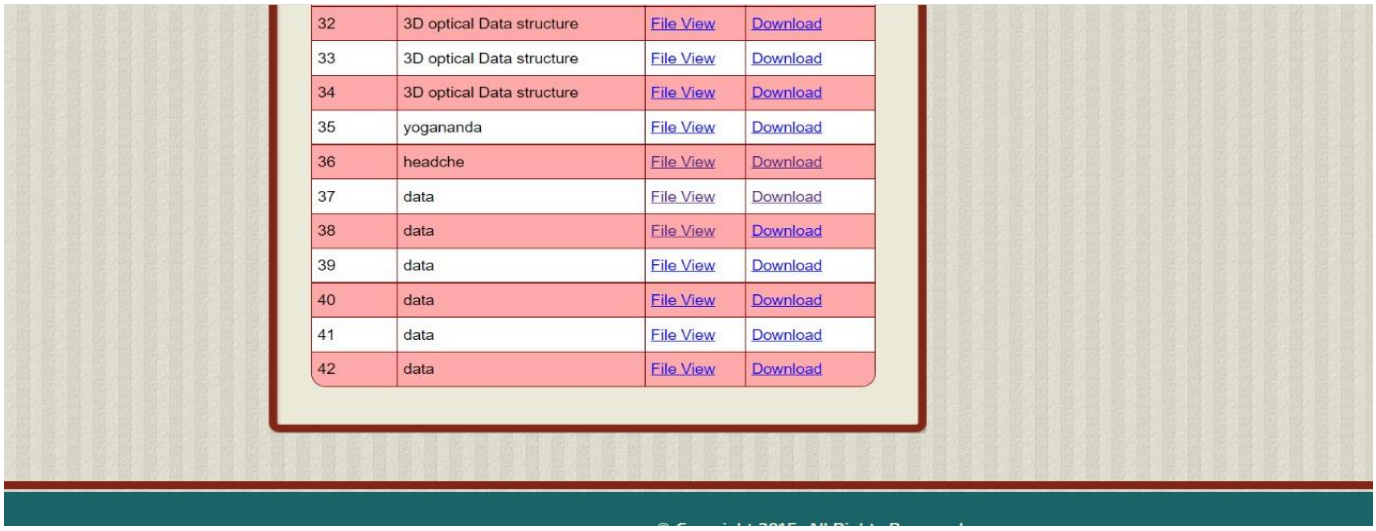
Screen 11.15. Auditor Public Key

25	3D optical Data structure	File View	Download
26	3D optical Data structure	File View	Download
27	3D optical Data structure	File View	Download
28	3D optical Data structure	File View	Download
29	3D optical Data structure	File View	Download
30	3D optical Data structure	File View	Download
31	3D optical Data structure	File View	Download
32	3D optical Data structure	File View	Download
33	3D optical Data structure	File View	Download
34	3D optical Data structure	File View	Download
35	yogananda	File View	Download
36	headche	File View	Download
37	data	File View	Download

Screen 11.16. Auditor File View



Screen 11.17. Auditor File View Data



32	3D optical Data structure	File View	Download
33	3D optical Data structure	File View	Download
34	3D optical Data structure	File View	Download
35	yogananda	File View	Download
36	headche	File View	Download
37	data	File View	Download
38	data	File View	Download
39	data	File View	Download
40	data	File View	Download
41	data	File View	Download
42	data	File View	Download

Screen 11.18. Auditor Download

Screen 11.19. Auditor File download

12. CONCLUSION

In this project, we addressed the problem of securing data outsourced to the cloud against an adversary which has access to the encryption key. For that purpose, we introduced a novel security definition that captures data confidentiality against the new adversary. We then proposed Bastion, a scheme which ensures the confidentiality of encrypted data even when the adversary has the encryption key, and all but two ciphertext blocks. Bastion is most suitable for settings where the ciphertext blocks are stored in multi-cloud storage systems. In these settings, the adversary would need to acquire the encryption key, and to compromise all servers, in order to recover any single block of plaintexts. Bastion considerably improves the performance of existing primitives which offer comparable security under key exposure, and only incurs a negligible overhead when compared to existing semantically secure encryption modes.

13. FUTURE SCOPE

The future scope of this project is to provide support for multimedia files such as image, audio and video. Our Project provides an environment where a data owner can share text data content with members of his group while preventing any outsiders gaining any data access in case of any malicious activities such as key theft.

REFERENCES

BOOKS REFERRED

[1] Software Engineering by Roger S Pressman

[2] OOAD by Grady Booch

[3] JAVA by Herbert Schildt

[4] Cloud Computing by K. Chandrasekaran

[5] Cryptography and Network Security by William Stallings

PAPERS REFERRED

1. M. Abd-El-Malek, G. R. Ganger, G. R. Goodson, M. K. Reiter, and J. J. Wylie, "Fault-Scalable Byzantine Fault-Tolerant Services," in ACM Symposium on Operating Systems Principles (SOSP), 2005, pp. 59–74.
2. M. K. Aguilera, R. Janakiraman, and L. Xu, "Using Erasure Codes Efficiently for Storage in a Distributed System," in International Conference on Dependable Systems and Networks (DSN), 2005, pp. 336–345.
3. W. Aiello, M. Bellare, G. D. Crescenzo, and R. Venkatesan, "Security amplification by composition: The case of doubly iterated, ideal ciphers," in Advances in Cryptology (CRYPTO), 1998, pp. 390– 407.
4. C. Basescu, C. Cachin, I. Eyal, R. Haas, and M. Vukolic, "Robust Data Sharing with Key-value Stores," in ACM SIGACT SIGOPS Symposium on Principles of Distributed Computing (PODC), 2011, pp. 221–222.
5. A. Beimel, "Secret-sharing schemes: A survey," in International Workshop on Coding and Cryptology (IWCC), 2011, pp. 11–46.
6. A. Bersani, M. Correia, B. Quaresma, F. André, and P. Sousa, "Dempski: Dependable and Secure Storage in a Cloud-of clouds," in Sixth Conference on Computer Systems (Ersoys), 2011, pp. 31–46.
7. G. R. Blakley and C. Meadows, "Security of ramp schemes," in Advances in Cryptology (CRYPTO), 1984, pp. 242–268.
8. V. Boyko, "On the Security Properties of OAEP as an All Or-nothing Transform," in in Proceedings of CRYPTO, 1997.

9. C. Charnes, J. Pietrzyk, and R. Safavi-Naini, “Conditionally secure secret sharing schemes with disenrollment capability,”
10. A. Desai, “The security of all-or-nothing encryption: Protecting against exhaustive key search,” in Advances in Cryptology (CRYPTO), 2000, pp. 359–375.
11. C. Dubicki, L. Grys, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczyk, J. Szczepkowski, C. Ungureanu, and M. Welinski, “Hydrator: a Scalable Secondary .
12. M. Demuth and D. M. Freeman, “Deniable encryption with negligible detection probability: An interactive construction,” in EUROCRYPT, 2011, pp. 610–626
13. EMC, “Transform to a Hybrid Cloud,” http://www.emc.com/campaign/global/hybrid cloud/index.htm.
14. ITIF. How Much Will PRISM Cost the U.S. Cloud Computing Industry? http://www2. itif.org/2013- cloud-computing-costs.pdf.
15. M. Klonowski, P. Kubiak, and M. Kutylowski. Practical Deniable Encryption. In Proceedings of SOFSEM: Theory and Practice of Computer Science, 2008.