

**DATA LEAKAGE DETECTION IN
CLOUD COMPUTING ENVIRONMENT
PROJECT REPORT
21AD1513- INNOVATION PRACTICES LAB**

Submitted by

**SWATHI P Reg. No. 211422243326
THAMARAISELVI S Reg. No. 211422243333**

in partial fulfillment of the requirements for the award of degree

of

**BACHELOR OF TECHNOLOGY
in
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**



PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123

ANNA UNIVERSITY: CHENNAI-600 025

November, 2024

BONAFIDE CERTIFICATE

Certified that this project report titled “**DATA LEAKAGE DETECTION IN CLOUD COMPUTING ENVIRONMENT**” is the bonafide work of **SWATHI P (211422243326)**, **THAMARAISELVI S (211422243333)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

INTERNAL GUIDE

Dr. K. JAYASHREE,M.E., Ph.D,
Professor,
Department of AI &DS,
Panimalar Engineering College,
Chennai-600123.

HEAD OF THE DEPARTMENT

Dr.S.MALATHI M.E., Ph.D
Professor and Head,
Department of AI & DS,
Panimalar Engineering College,
Chennai-600123.

Certified that the candidate was examined in the Viva-Voce Examination held on
.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ABSTRACT

In the recent years internet technologies has become the backbone of any business organization. These organizations use this facility to improve their efficiency by transferring data from one location to another. But there are number of threats in transferring critical organizational data as any culprit employee may public this data. This problem is known as data leakage problem. In the proposed work, we are suggesting a model for data leakage problem. In this model, our aim is to identify the culprit who has leaked the critical organizational data. Data leakage detection in cloud computing is an essential research area due to the increasing usage of cloud computing services. As more and more data is being stored and processed on the cloud, it becomes crucial to detect any data leakage or unauthorized access to prevent data breaches.

The objective of this research paper is to propose a data leakage detection mechanism in cloud computing environments. The proposed mechanism utilizes a hybrid approach that combines both static and dynamic analysis techniques to detect data leakage. Static analysis is performed on the source code to identify potential data leakage points, whereas dynamic analysis is carried out during the runtime to detect any actual data leakage.

The proposed mechanism also utilizes machine learning algorithms to improve the accuracy of data leakage detection. The machine learning algorithms are trained on the features extracted from the static and dynamic analysis, which enables the system to identify patterns and anomalies in the data that may indicate data leakage.

ACKNOWLEDGEMENT

A project of this magnitude and nature requires the kind cooperation and support of many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

We would like to express our deep gratitude to Our **Beloved Secretary and Correspondent, Dr.P. CHINNADURAI, M.A., Ph.D.**, for his kind words and enthusiastic motivation which inspired us a lot in completing the project.

We also express our sincere thanks to Our **Dynamic Directors Mrs.C. VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR, M.E., Ph.D.**, and **Dr. S. SARANYA SREE SAKTHIKUMAR, B.E., M.B.A., Ph.D.**, for providing us with the necessary facilities for the completion of this project.

We would like to express thanks to our **Principal, Dr. K. MANI, M.E., Ph.D.**, for having extended his guidance and cooperation.

We would also like to thank our **Head of the Department, Dr. S. MALATHI, M.E., Ph.D.**, of Artificial Intelligence and Data Science for her encouragement.

Personally, we thank our Supervisor **Mrs.Dr. K. JAYASHREE, M.E., Ph.D.** Assistant Professor, Department of Artificial Intelligence and Data Science for the persistent motivation and support for this project, who at all times was the mentor of germination of the project from a small idea.

We express our thanks to the project coordinators **Dr. A. JOSHI, M.E., Ph.D.**, Professor, **Ms. Hilda Jerlin, M.E.**, Assistant Professor in the Department of Artificial Intelligence and Data Science for their Valuable suggestions from time to time at every stage of our project.

Finally, we would like to take this opportunity to thank our family members, friends, and well-wishers who have helped us for the successful completion of our project.

We also take the opportunity to thank all faculty and non-teaching staff members in our department for their timely guidance in completing our project.

SWATHI P
(211422243326)

THAMARAISELVI S
(211422243333)

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	
	1.1 Project Description	1
	1.2 Objective	2
2	LITERATURE SURVEY	4
3	SYSTEM ANALYSIS	
	3.1 Introduction	7
	3.2 Feasibility Study	7
	3.2.1 Technical feasibility	7
	3.2.2 Social feasibility	8
	3.2.3 Economical feasibility	8
	3.3 Existing System	9
	3.3.1 Disadvantages	9
	3.4 Proposed System	9
	3.4.1 Advantages	10
	3.5 System Requirements	10
	3.5.1 Hardware Requirements	10
	3.5.2 Software Requirements	10
	3.6 Language Specification	10
	3.6.1 Features of Java	10

4	SYSTEM DESIGN	
	4.1 Data Flow Diagram	26
	4.2 UML Diagrams	28
	4.2.1 Use case Diagram	28
	4.2.2 Sequence Diagram	30
5	SYSTEM DEVELOPMENT	
	5.1 Module Description	32
6	SYSTEM TESTING	
	6.1 Introduction	35
	6.2 Objectives of Testing	35
	6.3 Types of Testing	35
	6.4 Testing Strategies	36
7	SYSTEM IMPLEMENTATION	38
8	SYSTEM MAINTENANCE	
	8.1 Introduction	39
	8.1.1 Corrective maintenance	39
	8.1.2 Adaptive maintenance	39
	8.1.3 Perfective maintenance	39
	8.1.4 Preventive maintenance	40
9	CONCLUSION	41
10	FUTURE ENHANCEMENT	42
	APPENDICES	
	Appendix 1 – Screen Shots	43
	Appendix 2 – Sample Coding	46
	REFERENCES	51

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1	Architectural diagram	1
2	Java Framework	9
3	Overall Data Flow Diagram	26
4	Use case Diagram	27
5	Sequence Diagram	28

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
DFD	DATA FLOW DIAGRAM
AIC	AVAILABILITY INTERGRITY AND CONFIDENTIALITY
UML	UNIFIED MODELING LANGUAGE
GUI	GRAPHICAL INTERFACE
TCP	TRANSMISSION CONTROL PROTOCOL
UDP	USER DATAGRAM PROTOCOL
IP	INTERNET PROTOCOL
FCE	FUZZY COMPREHENSIVE EVALUATION

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPTION

In the current business scenario, data leakage is a big challenge as critical organizational data should be protected from unauthorized access. Data leakage may be defined as the accidental or intentional distribution of private organizational data to the unauthorized entities. It is important to protect the critical data from being misused by any unauthorized use. Critical data include intellectual copy right information, patent information, functional information etc. In many organizations, this critical organizational data have been shared to many stakeholder outside the organizational premises. Therefore, it is difficult to identify the culprit, who has leaked the data[1][2]. In the proposed work, our goal is to identify the guilty user when the organizational data have been leaked by some agent. In the proposed work, Bell-La Padula security model has been used which provide the analysis and design of secure computer systems. This model is called data confidentiality model. Bell-LaPadula model mainly focuses on data confidentiality issues and provides controlled access to classified information. In contrast to the Biba-Integrity model which describes rule for the protection of data integrity[3]. In this formal model, the entities in an information system are divided into subjects and objects. The notion of a "secure state" is defined, and it is proven that each state transition preserves security by moving from one secure state to other secure state, thereby inductively proving that the system satisfies the security objectives of the model. The Bell-LaPadula model is built on the concept of a state machine with a set of allowable states in a computer system. A system state is defined to be secure if the only permitted access modes of subjects to objects are in accordance with a security policy. Cloud computing has become increasingly popular due to its ability to provide on-demand computing resources and cost-effective solutions. However, with the increasing usage of cloud computing services, the risk of data breaches and data leakage has also increased. Data leakage refers to the unauthorized disclosure of confidential or sensitive information, which can lead to significant financial and reputational damage for individuals and organizations.

Data leakage detection is crucial in cloud computing environments to prevent such incidents from occurring. The detection of data leakage can be challenging in cloud computing environments due to the distributed nature of cloud computing, where data is stored and processed in various locations. Additionally, the traditional security mechanisms used in non-cloud environments may not be sufficient

to protect against data leakage in cloud computing. In recent years, research has focused on developing new techniques and mechanisms for detecting data leakage in cloud computing environments. These mechanisms utilize various techniques such as static analysis, dynamic analysis, and machine learning algorithms to identify potential data leakage points and detect actual data leakage during runtime.

Static analysis involves analyzing the source code of applications to identify potential data leakage points. Dynamic analysis, on the other hand, involves monitoring the behavior of applications during runtime to detect any actual data leakage. Machine learning algorithms are used to improve the accuracy of data leakage detection by identifying patterns and anomalies in the data that may indicate data leakage.

1.1 CLOUD COMPUTING

Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centres available to many users over the Internet. A simple definition of cloud computing involves delivering different types of services over the Internet. From software and analytics to secure and safe data storage and networking resources, everything can be delivered via the cloud. You can access it from just about any computer that has internet access.

1.2 NETWORK CODING TECHNIQUES

In a network coding protocol, each intermediate node (except sender/receiver nodes) on a network path combines incoming packets to output another packet. These protocols enjoy higher throughput, efficiency and scalability than the store-and-forward routing, but they are prone to pollution attacks by malicious intermediate nodes injecting invalid packets. These packets produce more such packets downstream, and the receiver might not finally decode the file sent by the sender node. Secure network coding (SNC) protocols use cryptographic techniques to prevent these attacks: the sender authenticates each packet by attaching a small tag to it. These authentication tags are generated using homomorphic message authentication codes (MACs) or homomorphic signatures. Due to homomorphic property, an intermediate node can combine incoming packets (and their tags) into a packet and its tag in particular, they show that one can exploit some of the algorithms involved in an SNC protocol in order to construct a secure cloud storage protocol for static data. However, their construction does not handle dynamic data — that makes it insufficient in many applications where a client needs to update (insert, delete or modify) the remote data efficiently. Further investigations are needed towards an efficient DSCS construction using a secure network coding (SNC) protocol.

Network coding techniques have been used to construct distributed storage systems where the client's data are disseminated across multiple servers. However, they primarily aim to reduce the repair bandwidth when some of the servers fail. On the other hand, we explore whether we can exploit the algorithms involved in an SNC protocol to construct an efficient and secure cloud storage protocol for dynamic data (for a single storage server). Although dynamic data are generic in the sense that they support arbitrary update (insertion, deletion and modification) operations, append-only data (where new data corresponding to a data file are inserted only at the end of the file) find numerous applications as well. These applications primarily maintain archival as well as current data by appending the current data to the existing datasets. Examples of append-only data include data obtained from CCTV cameras, ledgers containing monetary transactions, medical history of patients, data stored at append-only databases, and so on. Append-only data are also useful for maintaining other log structures (e.g., certificates are stored using append-only log structures in certificate transparency schemes). In many of such applications, the data owner requires a cloud server to store the bulk data in an untampered and retrievable fashion with append being the only permissible update. Although secure cloud storage schemes for generic dynamic data also work for append-only data, a more efficient solution (specific to append-only data files) would be helpful in this scenario.

CHAPTER 2

LITERATURE SURVEY

1. TITLE: Publicly verifiable secure cloud storage for dynamic data using secure network coding

Author: B. Sengupta and S. Ruj

Cloud service providers offer storage outsourcing facility to their clients. In a secure cloud storage (SCS) protocol, the integrity of the client's data is maintained. In this work, we construct a publicly verifiable secure cloud storage protocol based on a secure network coding (SNC) protocol where the client can update the outsourced data as needed. To the best of our knowledge, our scheme is the first SNC-based SCS protocol for dynamic data that is secure in the standard model and provides privacy-preserving audits in a publicly verifiable setting. Furthermore, we discuss, in details, about the (im)possibility of providing a general construction of an efficient SCS protocol for dynamic data (DSCS protocol) from an arbitrary SNC protocol. In addition, we modify an existing DSCS scheme (DPDP I) in order to support privacy-preserving audits. We also compare our DSCS protocol with other SCS schemes (including the modified DPDP I scheme). Finally, we figure out some limitations of an SCS scheme constructed using an SNC protocol.

2. TITLE: MAC-based integrity for network coding. In Applied Cryptography and Network Security

Author: S. Agrawal and D. Boneh

Network coding has been shown to improve the capacity and robustness in networks. However, since intermediate nodes modify packets en-route, integrity of data cannot be checked using traditional MACs and checksums. In addition, network coded systems are vulnerable to pollution attacks where a single malicious node can flood the network with bad packets and prevent the receiver from decoding the packets correctly. Signature schemes have been proposed to thwart such attacks, but they tend to be too slow for online per-packet integrity. Here we propose *a* homomorphic MAC which allows checking the integrity of network coded data. Our homomorphic MAC is designed as a drop-in replacement for traditional MACs (such as HMAC) in systems using network coding.

3. TITLE: Enabling public auditability and data dynamics for storage security in cloud computing

Author: Q. Wang, C. Wang, K. Ren

Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of the client through the auditing of whether his data stored in the cloud is indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data dynamics via the most general forms of data operation, such as block modification, insertion and deletion, is also a significant step toward practicality, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lacks the support of either public auditability or dynamic data operations, this paper achieves both. We first identify the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for the seamless integration of these two salient features in our protocol design.

4. TITLE: Scalable and efficient provable data possession

Author: G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik

Storage outsourcing is a rising trend which prompts a number of interesting security issues, many of which have been extensively investigated in the past. However, Provable Data Possession (PDP) is a topic that has only recently appeared in the research literature. The main issue is how to frequently, efficiently and securely verify that a storage server is faithfully storing its client's (potentially very large) outsourced data. The storage server is assumed to be untrusted in terms of both security and reliability. (In other words, it might maliciously or accidentally erase hosted data; it might also relegate it to slow or off-line storage.) The problem is exacerbated by the client being a small computing device with limited resources. Prior work has addressed this problem using either public key cryptography or requiring the client to outsource its data in encrypted form. In this paper, we construct a highly efficient and provably secure PDP technique based entirely on symmetric key cryptography, while not requiring any bulk

encryption. Also, in contrast with its predecessors, our PDP technique allows outsourcing of dynamic data, i.e, it efficiently supports operations, such as block modification, deletion and append.

5.TITLE: DD-POR: Dynamic operations and direct repair in network coding-based proof of retrievability.

Author: K. Omote and T. T. Phuong

POR (Proof of Retrievability) is a protocol by which clients can distribute their data to cloud servers and can check if the data stored in the servers is available and intact. Based on the POR, the network coding is applied to improve network throughput. Although many network coding-based PORs have been proposed, most of them have not achieved the following practical features: direct repair and dynamic operations. In this paper, we propose the DD-POR (Dynamic operations and Direct repair in network coding-based POR) to address these shortcomings. When a server is corrupted, the DD-POR can support the direct repair in which the data stored in the corrupted server can be repaired using the data provided directly from the healthy servers. The client is thus free from the burden of data repair. Furthermore, the DD-POR allows the client to efficiently perform dynamic operations, i.e., modification, insertion and deletion.

CHAPTER 3

SYSTEM ANALYSIS

3.1 INTRODUCTION

System analysis refers to the study of existing system in terms of system goals. The system analysis of a project includes the basic analysis for the project development, the required data to develop the project, the cost factor considered for the project development and other related factors.

3.2 FEASIBILITY STUDY

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

- To analyze whether the software will meet organizational requirements
- To determine whether the software can be implemented using the current technology and within the specified budget and schedule
- To determine whether the software can be integrated with other existing software

Types of Feasibility

Commonly considered feasibility includes:

3.2.1 Technical Feasibility

3.2.2 Social Feasibility

3.2.3 Economic Feasibility

3.2.1 Technical Feasibility

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

Analyzes the technical skills and capabilities of the software development team members Determines whether the relevant technology is stable and established ascertains that the technology chosen for software development has a large number of users so that they can be consulted when problems arise or improvements are required.

3.2.2 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.

3.2.3 Economic Feasibility

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

- Cost incurred on software development to produce long-term gains for an organization.
- Cost required to conduct full software investigation such as, requirements elicitation and requirements analysis.
- Cost of hardware, software, development team, and training.

3.3 EXISTING SYSTEM

The existing system for data leakage detection in cloud computing environment typically involves various tools and techniques aimed at identifying and preventing data breaches. These include:

1. **Encryption:** Data encryption is one of the most common methods used to protect data in the cloud. It involves encoding data in such a way that it becomes unreadable without a key.
2. **Access Control:** Access control mechanisms such as firewalls, intrusion detection systems (IDS), and intrusion prevention systems (IPS) are used to control who can access the data stored in the cloud.
3. **Auditing:** Regular auditing and monitoring of the cloud environment can help detect and prevent data leakage. It involves reviewing logs and other data sources for any suspicious activity.
4. **Data Loss Prevention (DLP) Systems:** These systems are designed to detect and prevent data leakage by monitoring data in transit and at rest.
5. **Two-factor Authentication:** This involves requiring users to provide two forms of authentication, such as a password and a security token, before they can access data in the cloud.

Virtual Private Network (VPN): VPNs can be used to provide a secure connection between the user's device and the cloud environment, preventing unauthorized access.

3.3.1 DISADVANTAGES OF EXISTING SYSTEM:

- Cannot store the large volume of data and computational is less.
- Malicious data can attack the client file.
- Less accuracy and the performance is low.

3.4 PROPOSED SYSTEM

Data leakage in a cloud computing environment can occur due to various reasons, including unauthorized access, hacking, and software vulnerabilities. Therefore, it is essential to have a robust data leakage detection system in place to ensure data security in cloud environments.

One proposed system for data leakage detection in cloud computing environments is a combination of machine learning algorithms and anomaly detection techniques. This system analyzes the patterns of user behavior and data access to detect any anomalies that may indicate potential data leakage. The system can detect abnormal user behavior, such as accessing data outside of normal working hours or downloading large amounts of data in a short time.

The system can also monitor network traffic and detect any unusual data transfers or communications. The system can use a combination of machine learning algorithms, such as clustering and classification, to identify potential data leakage events. The system can also employ statistical analysis techniques, such as regression and correlation analysis, to identify trends and patterns in data usage.

3.4.1 ADVANTAGES OF PROPOSED SYSTEM:

- The advantages of our scheme are proved for security performance, communication
- High security and more effective.
- User friendly and computation is more efficiency.
- Reliability of the data is more.
- User identity is not disclosed to the outside world.

3.5 SYSTEM REQUIREMENTS

3.5.1 HARDWARE REQUIREMENTS

- Processor : I5
- Speed : 2.2 GHz
- RAM : 2 GB
- Hard Disk : 160 GB

3.5.2 SOFTWARE REQUIREMENTS

- Platform : Windows7
- Front-end : HTML,CSS,JS
- Back-end : SQL Server 2008 R2

3.6 LANGUAGE SPECIFICATION

3.6.1 FEATURES OF JAVA

Java Technology

- Java technology is both a programming language and a platform.

The Java Programming Language

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

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

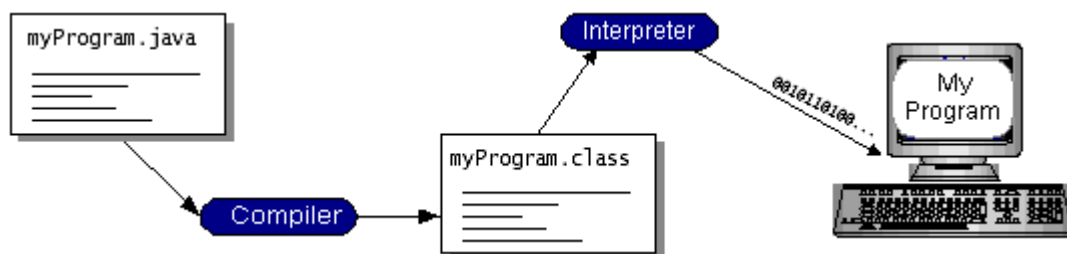


Figure NO: 1 Java Virtual Machine Diagram

It can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser

that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

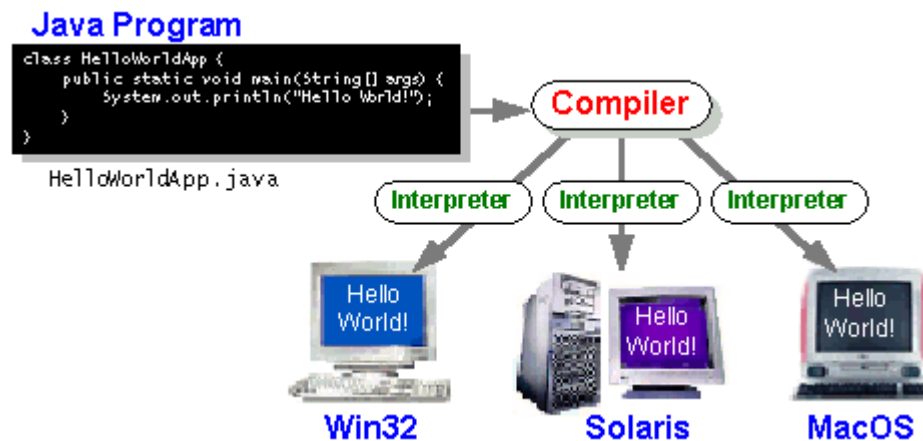


Figure No: 2 Java Program Diagram

The Java Platform

A platform is the hardware or software environment in which 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 on to various hardware-based platforms.

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

provide.

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

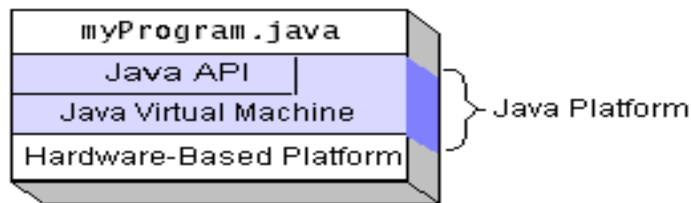


Figure No: 3 Java Platform Diagram

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

What Can Java Technology Do?

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

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

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server. How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

The essentials: Objects, strings, threads, numbers, input and output, data structures, system properties,

date and time, and so on.

Applets: The set of conventions used by applets.

Networking: URLs, TCP (Transmission Control Protocol), UDP (User Datagram 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.

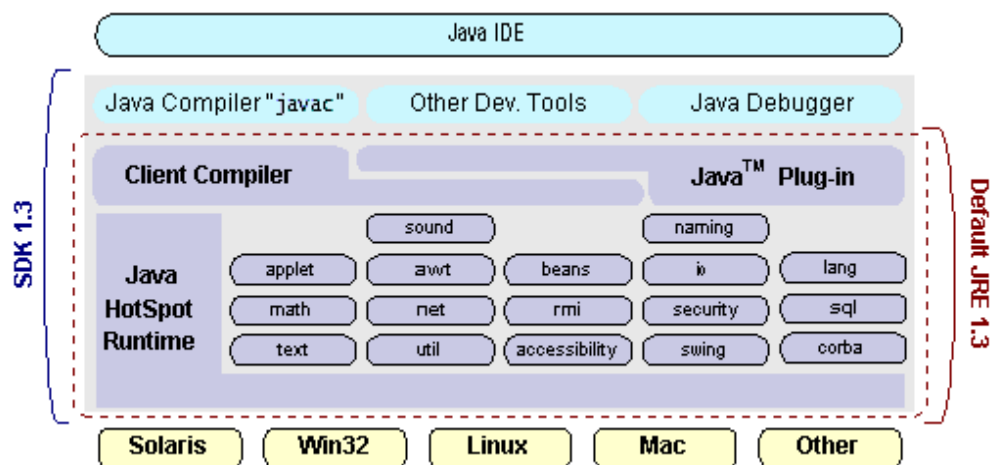


Figure No: 4 Java Database Connectivity (JDBC) Diagram

How Will Java Technology Change My Life?

It can't promise you fame, fortune, or even a job if you learn the Java programming language.

Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

Get started quickly: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.

Write less code: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.

Write better code: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.

Develop programs more quickly: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.

Avoid platform dependencies with 100% Pure Java: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.

Write once, run anywhere: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.

Distribute software more easily: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to.

Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs

suddenly change.

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

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

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even

Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true.

The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

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

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

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

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

JDBC Goals

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

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

1. SQL Level API

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

2. 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.

3. **JDBC must be implemental on top of common database interface**

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.

4. **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.

5. **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.

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

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

7. **Keep the common cases simple**

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

Finally, we decided to product the implementation using Java Networking.

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

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

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

Simple	Architecture-neutral
Object-oriented	Portable
Distributed	High-performance

Interpreted	multithreaded
Robust	Dynamic
Secure	

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

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

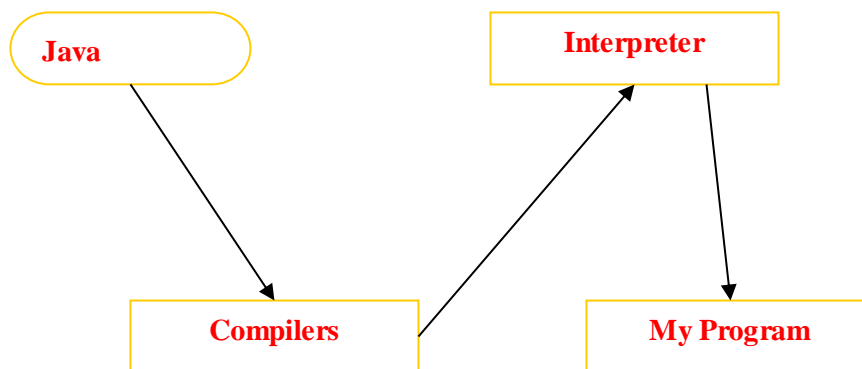


Figure No: 5 Java byte codes

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

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

Networking

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

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and

destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

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

TCP

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

Internet addresses

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

Network address

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

Subnet address

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

Host address

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

Total address

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

Port addresses

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

Sockets

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

```
#include <sys/types.h>
```

```
#include <sys/socket.h>
```

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

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

J Free Chart

J Free Chart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. J Free Chart'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);
- J Free Chart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licences (LGPL), which permits use in proprietary applications.

1. Map Visualizations

- Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:
- Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);
- Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XY Plot class in J Free Chart;

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

2. Time Series Chart Interactivity

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

3. Dashboards

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

1. Property Editors

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

J2ME (Java 2 Micro edition)

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems."

Announced in June 1999 at the Java One Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

1.	General	J2ME	architecture
-----------	----------------	-------------	---------------------

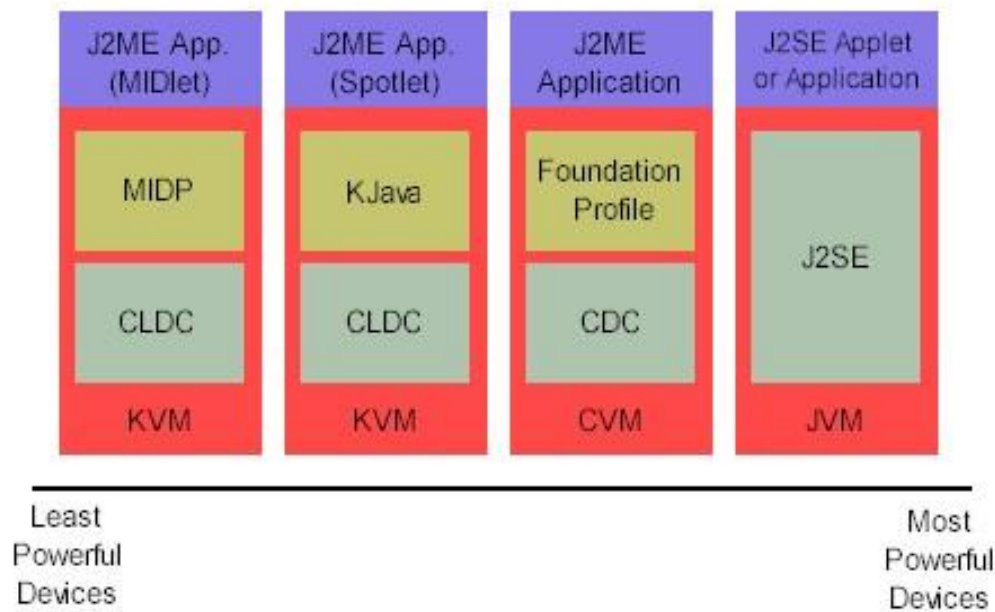


Figure No: 6 Java runtime Environment

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes.

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

2. Developing J2ME applications

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

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during

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

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

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

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

4. Configurations overview

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

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

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

5.J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The

Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: K Java and MIDP. Both K Java and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

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

Profile 1: K Java

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

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like K Java, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

* `java.lang`

* `java.io`

* `java.util`

* `javax.microedition.io`

* `javax.microedition.lcdui`

* `javax.microedition.midlet`

CHAPTER 4

SYSTEM DESIGN

SYSTEM ARCHITECTURE



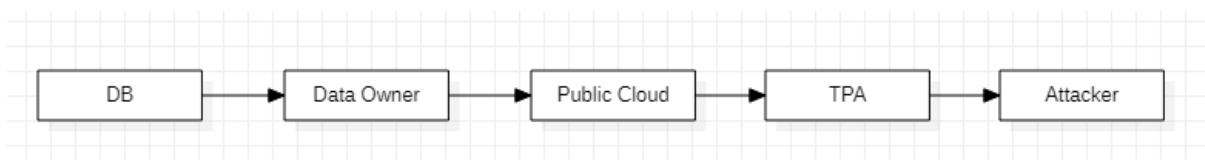
Figure No: 7 System Architecture

4.1 DATA FLOW DIAGRAM

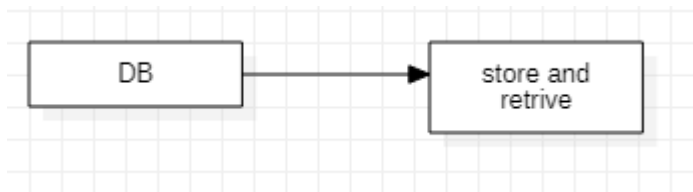
A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of the processing.

DATA FLOW DIAGRAM:

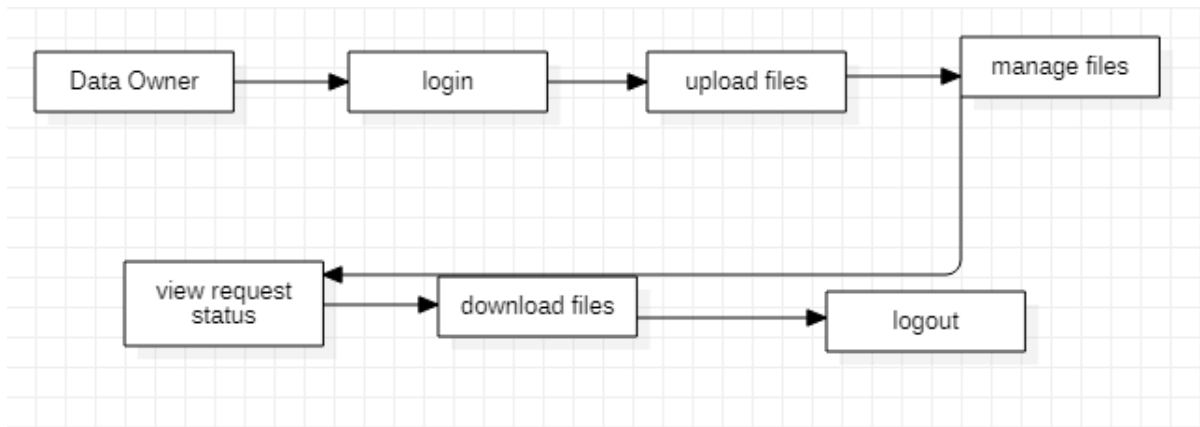
Level 0 :



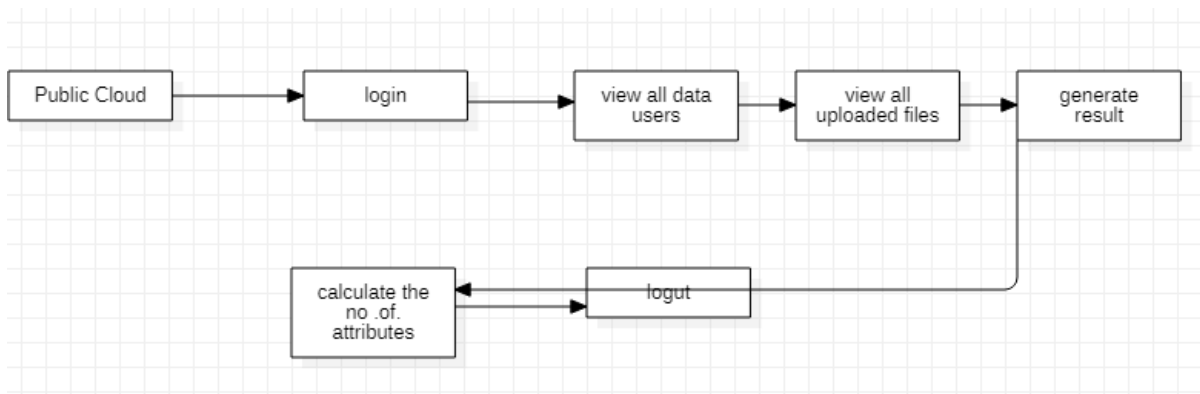
Level 1 :



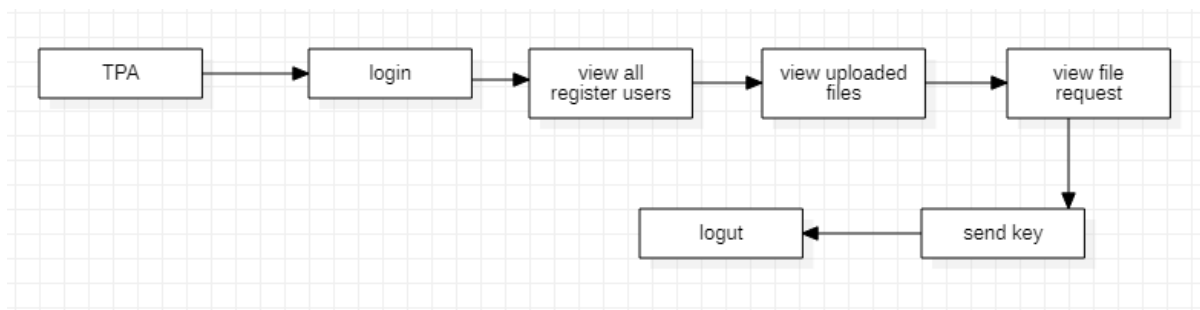
Level 2 :



Level 3 :



Level 4 :



Level 5 :

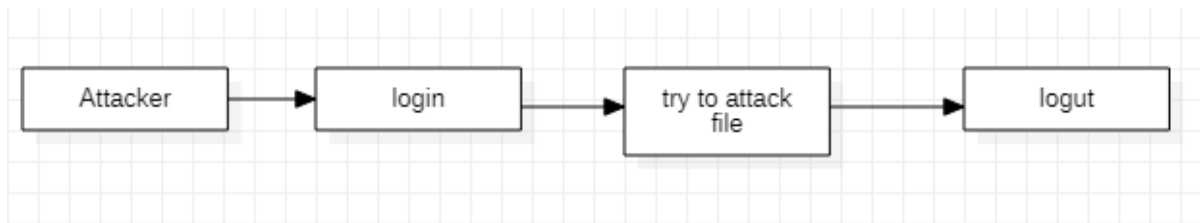


Figure No: 8 Data Flow Diagram

4.2 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 Objected 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 is comprised of 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 UML is a standard for specifying, Visualizing, Constructing and documenting the artifacts of software system, 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.


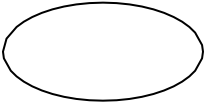
The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS OF UML

- Provides users ready-to-use, expressive visual modeling languages so that they can develop and exchange meaningful models.
- Provides extendibility and specification mechanisms to extend the core concepts.
- Be independent of particular programming language and development process.

4.2.1 USECASE DIAGRAM

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components a user or another system that will interact with the system modeled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.

Symbol	Meaning
	An actor . It specifies a role played by a user or any other system that interacts with the subject.
	A use case . It is a list of steps, typically defining interactions between an actor and a system, to achieve a goal.

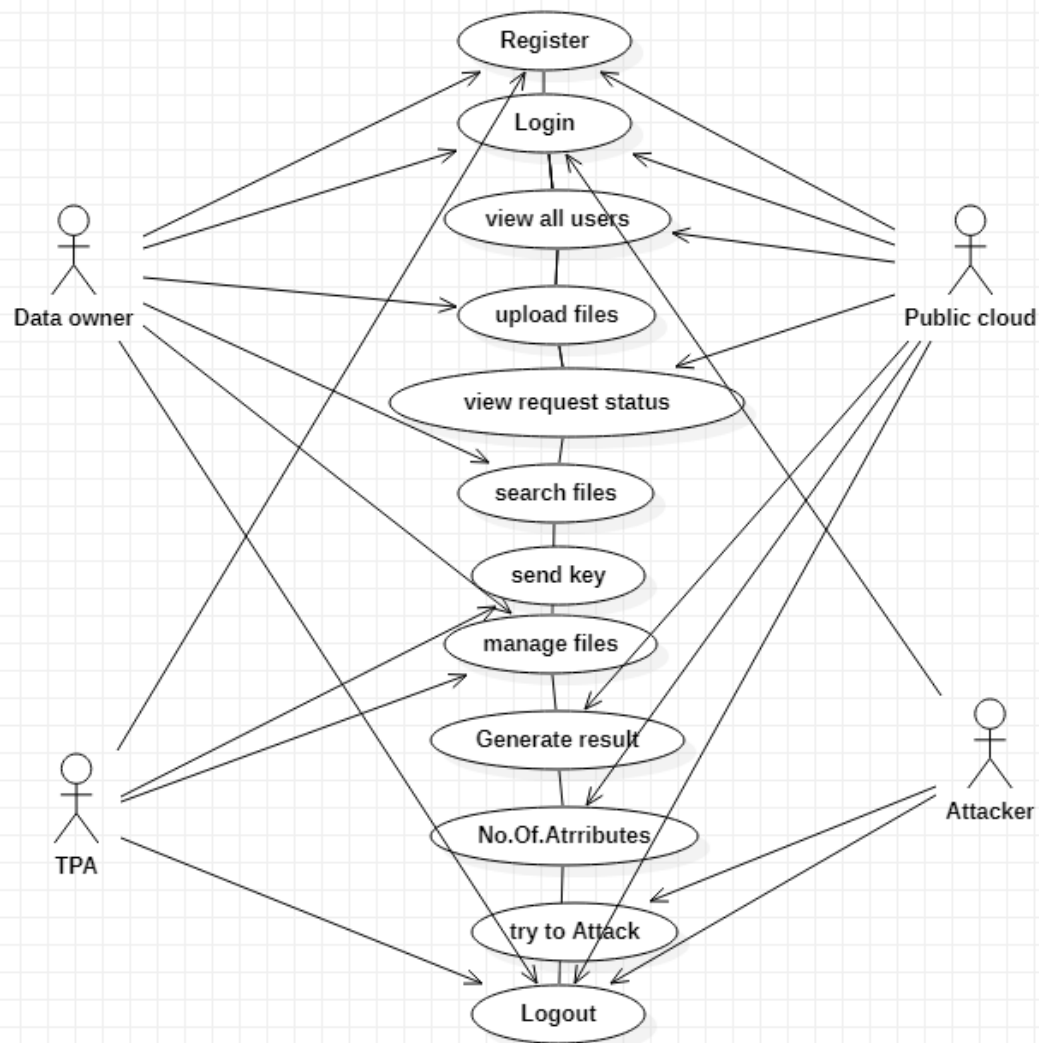

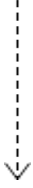




Figure No: 9 UML Diagram

4.2.2 SEQUENCE DIAGRAM

The sequence diagram shows how different objects interact with each and the order of those interactions occurs. They are ordered by time. They are useful if someone wants to review the flow of logic through a scenario.

Symbol	Meaning
	<p>An object represents a class or object, in UML. They demonstrate how an object will behave in the context of the system.</p>

	<p>Lifeline represents the passage of time as it extends downward.</p>
	<p>Activate is used to denote participant activation. Once a participant is activated, its lifeline appears.</p>
	<p>A Message is an element that defines a specific kind of communication between instances in an interaction</p>

The following symbols are used in a sequence diagram

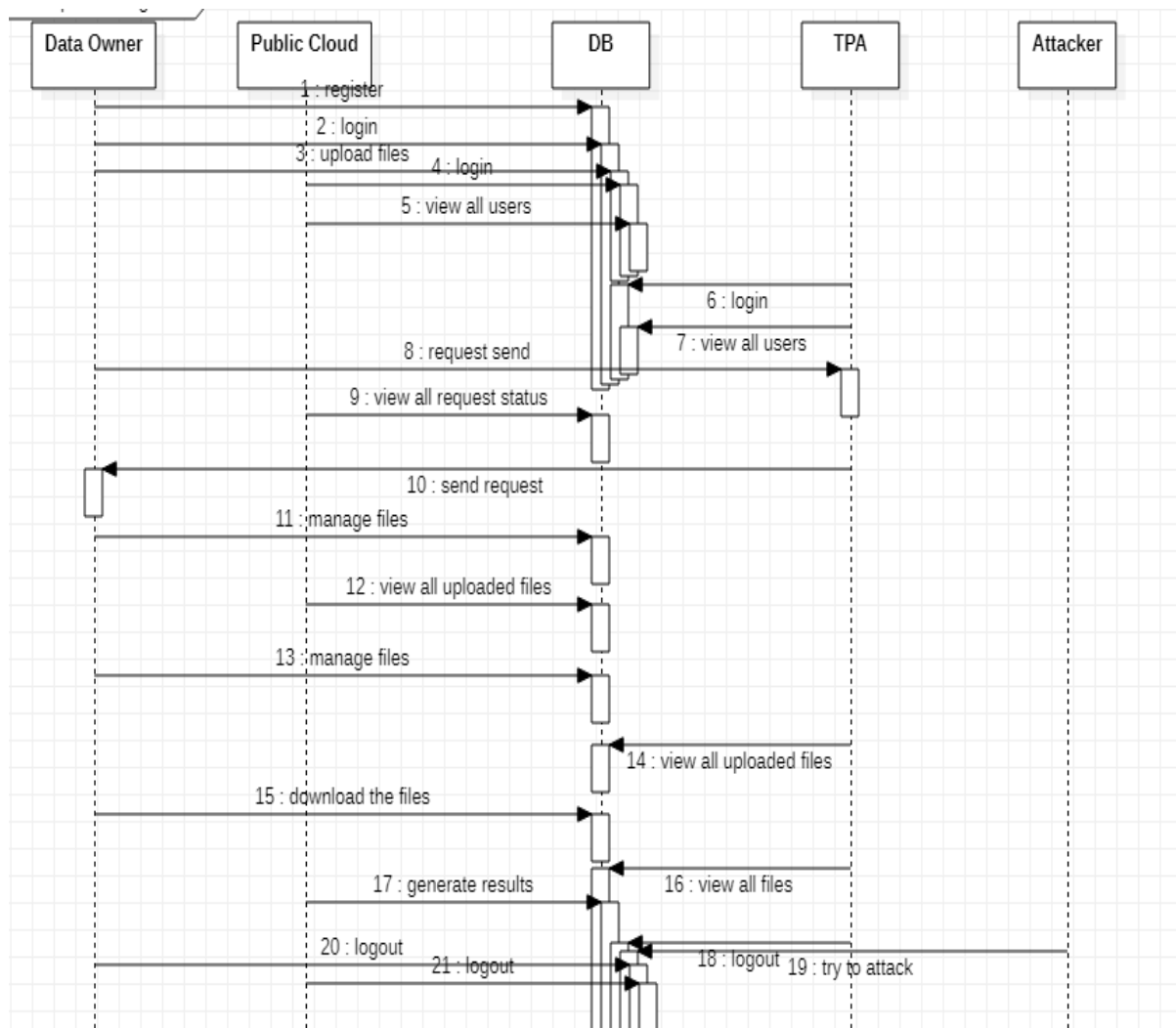


Figure No: 10 Sequence Diagram

CHAPTER 5

SYSTEM DEVELOPMENT

5.1 MODULE DESCRIPTION

In this project, we have four modules

1. Data Owner

2. TA (Trust Authority)

3. Cloud servers

4. Attackers

1. Data Owner

- Data Owner have to register first.
- Upload the files to the cloud in encrypted format with file private and trapdoor key by using fuzzy logic for key generation algorithm for encryption & decryption. The stored the cloud server.
- Manage the file.
- Search Files: User can search file with Encrypted format, then sends the request to the TA
- View Request Status Waiting or Accept.
- Download Files by using the file private key, user can download the files in decrypted format.
- Logout

2. TA (Trust Authority)

- Login Our account.
- View all registered Data Users.
- View all upload Files.
- View all user file request then all request backup is sent to cloud server.
- Logout

3. Cloud Server

- Login Our account.
- View all registered Data Users.
- View all upload Files.

- Result- Generate the result based on the file Storage backup.
- No. of. Attributes- calculate the number of attributes to the cloud.
- Logout

4. Attackers

- Login
- Try to attack the file.
- Logout

MODULES DESCRIPTION:

1. Data Owner

The process, data owners must first register on the platform. Once registered, they can upload their files to the cloud in an encrypted format. This involves using a fuzzy logic-based key generation algorithm to create both a file private key and a trapdoor key for secure encryption and decryption. After the files are securely stored on the cloud server, data owners can manage their files easily. Users also have the ability to search for files, even in their encrypted format. When a search request is initiated, it is sent to the Trusted Authority (TA) for processing. Users can view the status of their requests, which may be marked as "Waiting" or "Accepted." When the request is approved, users can download the files using their file private key, allowing them to access the files in a decrypted format. Finally, users can log out of the system, ensuring their session is securely terminated.

2. TA (Trust Authority)

To begin, TA can log into their accounts to access the platform. Once logged in, they can view a comprehensive list of all registered data users, allowing for easy management and oversight. Additionally, users can see all uploaded files, providing a clear overview of the available resources. TA also have the capability to view all file requests made by other users. To ensure data integrity and security, a backup of all these requests is automatically sent to the cloud server for safekeeping. When their tasks are complete, TA can securely log out of their accounts, ensuring that their session is properly terminated.

3. Cloud servers

To start, Cloud can log into their accounts to access the platform. Once logged in, they can view all registered data users, facilitating easy management and collaboration. Cloud can also browse through all uploaded files, which provides them with a clear understanding of the available data. From there,

Cloud can generate results based on the file storage backup, ensuring that they have up-to-date information on their resources. Additionally, they can calculate the number of attributes associated with the files stored in the cloud, offering valuable insights into the data structure. Once their tasks are complete, Cloud can securely log out of their accounts, ensuring that their session is terminated and their data remains protected.

4. Attackers

The attackers log into their account to perform file attacks and then log out when finished.

CHAPTER 6

SYSTEM TESTING

6.1 INTRODUCTION

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

6.2 OBJECTIVES OF TESTING

Main objective of testing is to find errors. The success of testing in revealing errors in programs depends on the critical test cases. A good test case is one that has a high probability of finding an as-yet-undiscovered error or fault. A successful testing is one that reveals undiscovered errors to make the software more rugged and reliable.

Errors can be injected at any stage during development. In each phase different techniques for detecting and eliminating errors are incorporated. Testing should be planned based on the complexity of the activity or work. The testing for this project has been done on an incremental basis, in which components and subsystems of the system are tested separately before integrating them to form the system for system testing.

6.3 TYPES OF TESTING

- White Box Testing
- Black Box Testing

6.3.1 WHITE BOX TESTING

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

6.3.2 BLACK BOX TESTING

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

The steps involved in black box test case design are

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

6.4 TESTING STRATEGIES

6.4.1 UNIT TESTING

It deals with the testing of the smallest units of the system design namely the module. This helps in identifying the syntax error/low-level logical errors at the lowest level possible. Unit testing positive result indicates that the module as such is error free and is limited to the boundary of the module itself. This means that the modules by it is error-free and may have errors when connected to other modules. This means that there may be massive logical errors, which can't be identified at the level of unit testing. The procedural design descriptions are helpful in testing the control paths within the boundary of the module. The unit testing methodology can be done to many modules of the same system simultaneously.

6.4.2 INTEGRATION TESTING

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

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

It involves testing of modules in series with regard to control flow system. This helps identifying the logical error that is the possible due to invalid number of arguments, invalid data type sent across the modules during the control flow. In other words, integration acts as a fine tune in finding out flaws in the system with more than one module in series. The approaches in integration testing namely, top-down approach give a better way to test the sub-system connected in series as in the control flow.

6.4.3 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.
System/Procedures	: interfacing system or procedures must be invoked.

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

6.4.4 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 strategy and approach

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

Test objectives

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

Features to be tested

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

CHAPTER 7

SYSTEM IMPLEMENTATION

Implementation is the most crucial stage in achieving a successful system and giving the user's confidence that the new system is workable and effective, implementation of modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the program specification, the computer system and its environment is tested to the satisfaction of the user.

The system that has been developed is accepted, proved to be satisfactory for the user, and so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions quickly.

Initially, as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to all users and the server is to be connected to a network. The final stage is to document the entire system which provides component and the operating procedure of the system. The implementation involves the following things:

- Proper planning
- Investigation of the system and constraints
- Design the method to achieve the changeover
- Training the methods to achieve the changeover
- Training of the staff in the change phase.

CHAPTER 8

SYSTEM MAINTENANCE

8.1 INTRODUCTION

In software maintenance, an enormous mass of potential problems and cost lies under the surface. Software maintenance is far more than fixing mistakes. Analyst and programmers append for more time in maintaining the program than they do writing them. Few tools and techniques are available for maintenance. The literature on maintenance contains very few entries when compared to development activities.

The software maintenance is classified into four tasks:

8.1.1 Corrective maintenance

8.1.2 Adaptive maintenance

8.1.3 Perfective maintenance

8.1.4 Preventive maintenance

8.1.1 CORRECTIVE MAINTENANCE

This type of maintenance implies removing errors in a program, which might have crept in the system due to faulty design or wrong assumptions. Thus, in corrective maintenance, processing or performance failures are required.

8.1.2 ADAPTIVE MAINTENANCE

In adaptive maintenance, program functions are changed to enable the information needs of the user. This type of maintenance may become necessary because of organizational changes which may include:

- Change in the organizational procedures,
- Change in organizational objectives, goals, policies, etc.
- Change in forms,
- Change in information needs of managers.
- Change in system controls and security needs, etc.

8.1.3 PERFECTIVE MAINTENANCE

Perfective maintenance means adding new programs or modifying the existing programs to enhance the performance of the information system. These types of maintenance are undertaken to respond to user's additional needs which may be due to the changes within or outside of the organization. Outside changes, primarily environmental changes, which may be in the absence of system maintenance, render the information system ineffective and inefficient. This environmental change includes:

- a) Changes in governmental policies, laws, etc.
- b) Economic and competitive conditions
- c) New technology

The results obtained from the evaluation process help the organization to determine whether its information systems are effective and efficient or otherwise.

8.1.4 PREVENTIVE MAINTENANCE

Preventive maintenance is a modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults.

CHAPTER 9

CONCLUSION

The proposed technique will provide better security against data leakage problem. We can detect the data leaker in real time by using this method. It also protect different types of active and passive attacks. The proposed technique is computationally cost effective in terms of time and space uses. Therefore, this can be useful in distributed computing environment to protect data from data leakage. The proposed technique is based on symmetric algorithm, therefore it is infeasible to extend this model for web environment where multiple number of users frequently accessing the data object. We can also implement this technique for asymmetric cryptography.

CHAPTER 10

FUTURE ENHANCEMENT

Detecting data leakage in cloud computing environments is an ongoing research area, and there are several promising avenues for future work. Here are some potential directions:

Develop more advanced machine learning algorithms: Machine learning algorithms can be trained to detect anomalous patterns in data access and transmission, which could indicate data leakage. However, current algorithms may not be effective enough in detecting complex data leakage scenarios. Future work could focus on developing more sophisticated machine learning algorithms that can detect more subtle patterns of data leakage.

Enhance privacy-preserving techniques: Techniques such as encryption, access control, and anonymization can be used to protect data in cloud computing environments. However, these techniques have limitations, such as performance overhead and the risk of insider attacks. Future work could focus on enhancing these privacy-preserving techniques to provide stronger protection against data leakage.

Develop better monitoring tools: Cloud computing environments generate vast amounts of data, and it can be challenging to monitor all data access and transmission activities. Future work could focus on developing better monitoring tools that can identify potential data leakage events in real-time.

Explore new detection techniques: Traditional detection techniques such as log analysis and network monitoring have limitations in detecting complex data leakage scenarios. Future work could explore new techniques such as behavior analysis and machine learning-based detection to improve the detection of data leakage.

Investigate the impact of emerging technologies: Emerging technologies such as blockchain, edge computing, and IoT are increasingly being used in cloud computing environments. Future work could investigate the impact of these technologies on data leakage detection and develop new techniques to address any new vulnerabilities that may arise.

SCREEN SHOTS



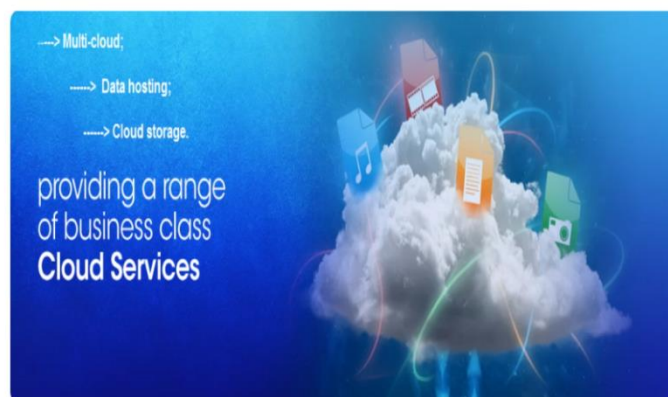
Menu ▾ Login / Register

Data Leakage Detection in Cloud Computing Environment



Data Leakage Detection in Cloud Computing Environment

- Home Page
- Upload File
- View Details
- Manage File
- View all File
- Logout



localhost:8080/SRC/AuUser.jsp

- Home Page
- View User & Authorized
- View Auditing Request
- Logout

View all User & Authorized them

Id	Name	DOB	Email	Contact	Address	Status
3	ram	21/01/2007	ram@gmail.com	9829828922	chenni	Authorized
8	logu	12/09/1999	camy@gmail.com	Chennai	chennai	Authorized
9	logu	12/09/1999	camy@gmail.com	Chennai	chennai	Authorized
10	guna	08-01-2001	guna@gmail.com	9786757689	chennai	Authorized

< 1 2 3 4 5 >

localhost:8080/SRC/Cloud.jsp

Data Leakage Detection in Cloud Computing Environment

Menu Login / Register

Cloud Login



Username *
cloud

Password * cloud

LOGIN

localhost:8080/SRC/Upload.jsp

- Home Page
- Upload File
- View Details
- Manage File
- View all File
- Logout

Upload file to guna

File key: 3747

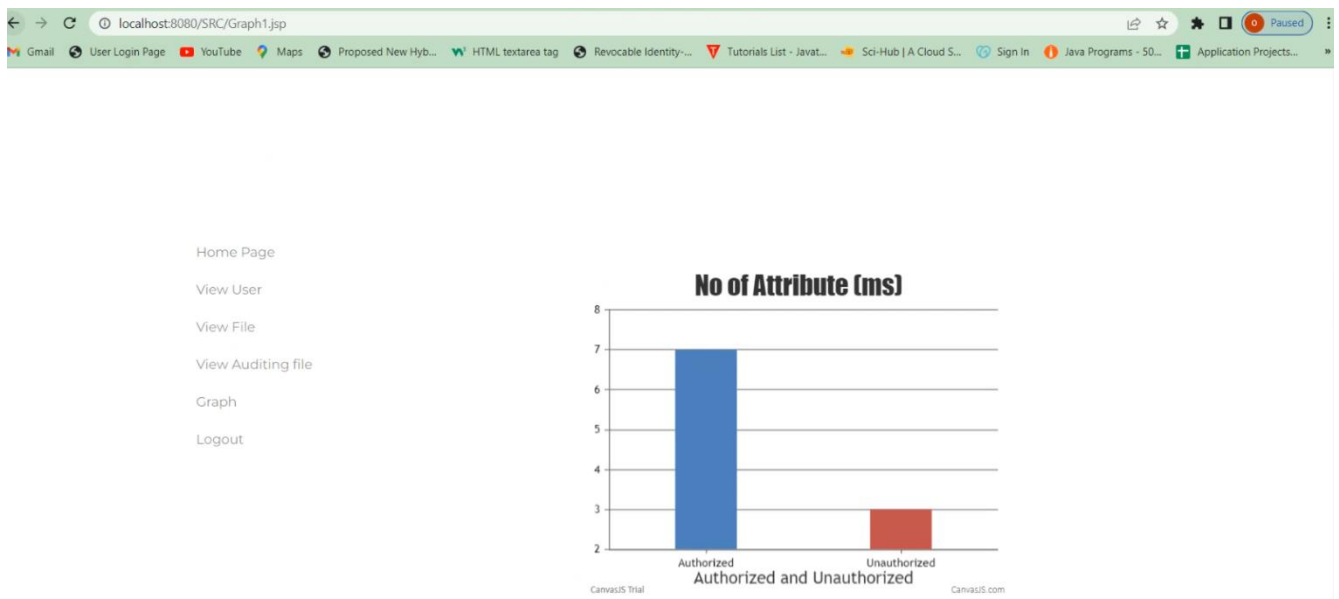
Date: 05/05/23 14:18:21

Trapdoor: 805BED230E57DB8E

Upload File: Choose File laptop.jpg

Upload

< 1 2 3 4 5 >



secure cloud storage with data dynamics using secure network coding techniques



Attacker Login

Username *

attacker

Password *

.....

LOGIN

APPENDIX 2

SAMPLE CODING

```

package dbServices;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DB {
private static final String URL "jdbc:mysql://localhost:3306/spam_filtering";
private static final String USER = "root";
private static final String PASSWORD = "root";
public Connection get Connection() throws SQL Exception {
return Driver Manager. get Connection(URL, USER, PASSWORD);
}
}
import java.io.IO Exception;
import java.io.Print Writer;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(urlPatterns = {"/Login_Action"})
public class Login_Action extends Http Servlet {
protected void process Request(Http Servlet Request request,
Http Servlet Response response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try(Print Writer out = response.getWriter()) {
String username = request.getParameter("username");
String password = request.getParameter("password");
if ("admin@gmail.com".equalsIgnoreCase(username) &&
"admin".equals(password)) {

out.println("<script>alert('Welcome Admin');</script>");

```

```

        Request Dispatcher rd = request.getRequestDispatcher("User_Home.jsp");
        rd.include(request, response);
    } else {
        out.println("<script>alert('Invalid Login');</script>");
        RequestDispatcher rd = request.getRequestDispatcher("index.jsp");
        rd.include(request, response);
    }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

protected void doGet(HttpServletRequest request,
HttpServletRequest response)
    throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request,
HttpServletRequest response)
    throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Handles user login";
}
}

import dbServices.DB;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;

```



```

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
public class movie extends HttpServlet {
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
    ServletContext sc = request.getSession().getServletContext();
    List<String> m = new ArrayList<>();
    String finalImage = "";
    boolean isMultipart =
ServletFileUpload.isMultipartContent(request);
    if (!isMultipart) {
        out.println("<script>alert('File Not Uploaded');</script>");
        return;
    }
    FileItemFactory factory = new DiskFileItemFactory();
    ServletFileUpload upload = new ServletFileUpload(factory);
    List<FileItem> items;
    try {
        items = upload.parseRequest(request);
    } catch (FileUploadException e) {
        e.printStackTrace();
        out.println("<script>alert('File upload failed.');"");
        return;
    }
    for (FileItem item : items) {
        if (item.isFormField()) {

```

```

        String value = item.getString();
        m.add(value);
    } else {
        String itemName = item.getName();
        finalImage = new File(itemName).getName();
        File savedFile = new File(sc.getRealPath("video") +
File.separator + finalImage);
        try {
            item.write(savedFile);
        } catch (Exception e) {
            e.printStackTrace();
            out.println("<script>alert('File save failed.');"</script>");
            return;
        }
    }
}

try (Connection con = new DB().getConnection();
        PreparedStatement ps = con.prepareStatement("INSERT
INTO movie(id, oname, file, skey, video, des) VALUES (?, ?, ?, ?, ?,
?)" )) {
    ps.setString(1, m.get(0));
    ps.setString(2, m.get(1));
    ps.setString(3, m.get(2));
    ps.setString(4, m.get(3));
    ps.setString(5, finalImage);
    ps.setString(6, m.get(4));

    ps.executeUpdate();

    out.println("<script>alert('File uploaded
successfully');"</script>");

    } catch (SQLException ex) {

        ex.printStackTrace();

        out.println("<script>alert('Database error
occurred.');"</script>");

    }

RequestDispatcher rd =

```

```
request.getRequestDispatcher("User_Home.jsp");

    rd.include(request, response);

}

}

protected void doGet(HttpServletRequest request,
HttpServletRequest response)

    throws ServletException, IOException {

    processRequest(request, response);

}

protected void doPost(HttpServletRequest request,
HttpServletRequest response)

    throws ServletException, IOException {

    processRequest(request, response);

}

public String getServletInfo() {

    return "Handles movie file upload";

}

}
```

REFERENCES

1. Rohit Pol, Vishwajeet Thakur, Ruturaj Bhise, and A Kat. Data leakage detection. *International Journal of Engineering Research & Application*, 2(3):404–410, 2012.
2. Rupesh Mishra and DK Chitre. Data leakage and detection of guilty agent. *International Journal of Scientific & Engineering Research*, 3(6), 2012.
3. Kenneth J Biba. Integrity considerations for secure computer systems. Technical report, DTIC Document, 1977.
4. David Elliott Bell. Bell–la padula model. *Encyclopedia of Cryptography and Security*, pages 74–79, 2011.
5. Mukesh Singhal and Niranjana G Shivaratri. *Advanced concepts in operating systems*. McGraw-Hill, Inc., 1994.
6. AL Jeeva, Dr V Palanisamy, and K Kanagaram. Comparative analysis of performance efficiency and security measures of some encryption algorithms. *International Journal of Engineering Research and Applications (IJERA) ISSN*, pages 2248–9622, 2012.
7. E Thambiraja, G Ramesh, and Dr R Umarani. A survey on various most common encryption techniques. *International journal of advanced research in computer science and software engineering*, 2(7):226–233, 2012.
8. Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. *Performance comparison of the aes submissions*, 1999.
9. Hamdan Alanazi, BB Zaidan, AA Zaidan, Hamid A Jalab, M Shabbir, Yahya Al-Nabhani, et al. New comparative study between des, 3des and aes within nine factors. *arXiv preprint arXiv:1003.4085*, 2010.
10. Aman Kumar, Sudesh Jakhar, and Sunil Makkar. Distinction between secret key and public key cryptography with existing glitches. *Indian Journal of Education and Information Management*, 1(9):392–395, 2012.
11. Hitendra GARG and Suneeta AGARWAL. A secure image based watermarking for 3d polygon mesh. *SCIENCE AND TECHNOLOGY*, 16(4):287–303, 2013.
12. Hitendra Garg and Suneeta Agrawal. Uniform repeated insertion of redundant watermark in 3d object. In *Signal Processing and Integrated Networks (SPIN), 2014 International Conference on*, pages 184–189. IEEE, 2014.
13. CISSP Susan Hansche, CISSP John Berti, and Chris Hare. *Official (ISC) 2 guide to the CISSP exam*. CRC Press, 2003.
14. D Elliott Bell and Leonard J La Padula. *Secure computer system: Unified exposition and multics interpretation*. Technical report, DTIC Document, 1976.
15. David Elliott Bell. Looking back at the bell-la padula model. In *ACSAC*, volume 5, pages 337–351, 2005.