ENHANCED DETECTION OF DATA LEAKAGE IN CLOUD COMPUTING ENVIRONMENT

A PROJECT REPORT

Submitted by

ANNIE S [211420104019]

ANNE FLORA R [211420104018]

SHARNIKA S A [211420104250]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report "**ENHANCED DATA LEAKAGE DETECTION IN CLOUD COMPUTING ENVIRONMENT** " is the bonafide work of " **ANNIE S(211420104019), ANNE FLORA R(211420104018), SHARNIKA S A(211420104250)**" who carried out the project work under my supervision.

Signature of the HOD with date	**Signature of the Supervisor with date**
DR L.JABASHEELA M.E., Ph.D.,	**Dr. T.TAMILVIZHI M.TECH., Ph.D.,**
PROFESSOR AND HEAD,	**SUPERVISOR, PROFESSOR,**
Department of Computer Science and Engineering,	**Department of Computer Science and Engineering,**
Panimalar Engineering College,	**Panimalar Engineering College,**
Chennai – 123.	**Chennai – 123.**

Certified that the above candidate(s) was examined in the End Semester Project Viva- Voce Examination held on .....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We "**ANNIE S(211420104019), ANNE FLORA R(211420104018), SHARNIKA S A(211420104250)**" hereby declare that this project report titled "**ENHANCED DATA LEAKAGE DETECTION IN CLOUD COMPUTING ENVIRONMENT**" , under the guidance of **Dr. TAMILVIZHI.T M.TECH., Ph.D** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ANNIE S(211420104019)

ANNE FLORA R(211420104018)

SHARNIKA S A(211420104520)

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D.,** for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project.

We want to express our deep gratitude to our Directors, **Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to Project Coordinator **Dr.KAVITHA SUBRAMANI M.E.,Ph.D** and **Dr.T.TAMILVIZHI M.TECH., Ph.D** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

ANNIE S

ANNE FLORA R

SHARNIKA S A

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 project 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.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

TPA	-	Third Party Auditor
JVM	-	Java Virtual Machine
CFA	-	Collaborative Filtering Algorithm

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

INTRODUCTION

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. 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. 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 PROBLEM STATEMENT

Our project's primary goal is to provide secure protection to the person using the resources and to identify the person attempting to access another person's data. Since someone can access another person's data through shared resources, we are preventing this situation and making the file safer by assigning unique keys to each piece of data. This way, whoever tries to access or attack the data will be unable to do so, because the admin will be monitoring all site activities. by providing distinct keys The information will be encrypted, making it unreadable to anyone without the key, Unless, if certainly, he owns the key, which is impossible.

1.2 SYSTEM STUDY

FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY
- ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the

research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

- TECHNICAL FEASIBILITY

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

- SOCIAL FEASIBILITY

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

CHAPTER 2
LITERATURE REVIEW

CHAPTER 2

LITERATURE SURVEY

PAPER 1:

Title: MLPAM: A Machine Learning and Probabilistic Analysis Based Model for Preserving Security and Privacy in Cloud Environment

Authors: Ishu Gupta, Member, Rishabh Gupta, Ashutosh Kumar Singh, Rajkumar Buyya.

Abstract: The organizational valuable data needs to be shared with multiple parties and stakeholders in a cloud environment for storage, analysis, and data utilization. However, to ensure the security, preserve privacy while sharing the data effectively among various parties have become formidable challenges. In this article, by utilizing encryption, machine learning, and probabilistic approaches, we propose a novel model that supports multiple participants to securely share their data for distinct purposes. The model defines the access policy and communication protocol among the involved multiple untrusted parties to process the owners' data. The proposed model minimizes the risk associated with the leakage by providing a robust mechanism for prevention coupled with detection. The experimental results demonstrate the efficiency of the proposed model for different classifiers over various datasets. The proposed model ensures high accuracy and precision up to 97% and 100% relatively and secures a significant improvement upto 0.01%, 103%, 151%, 87%, 96%, 43%, and 186% for average probability, average success rate, detection rate, accuracy, precision, recall, and specificity, respectively, compared to the prior works that prove its effectiveness.

PAPER 2:

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

Authors: B. Sengupta and S. Ruj, 2016

Abstract: 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.

PAPER 3:

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

Authors: S. Agrawal and D. Boneh. 2009

Abstract: Network coding has been shown to improve the capacity and robustness in networks. However, since intermediate nodes modify packets enroute, 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.

PAPER 4:

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

Authors: Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li,2011

Abstract: 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. In particular, to achieve efficient data dynamics, we improve the existing proof of storage models by manipulating the classic Merkle Hash Tree construction for block tag authentication. To support

efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed schemes are highly efficient and provably secure.

PAPER 5:

Title: Scalable and efficient provable data possession.

Authors: G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik,2009

Abstract: 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.

PAPER 6:

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

Authors: K. Omote and T. T. Phuong, 2015

Abstract: 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.

PAPER 7:

Title: Secure Data Sharing and Search for Cloud-Edge-Collaborative Storage.

Authors: Ye Tao, Peng Xu, Hai Jin

Abstract: Cloud-edge-collaborative storage (CECS) is a promising framework to process data of the internet of things (IoT). It allows edge servers to process IoT data in real-time and stores them on a cloud server. Hence, it can rapidly respond to the requests of IoT devices, provide a massive volume of cloud storage for IoT data, and conveniently share IoT data with users. However, due to the vulnerability of edge and cloud servers, CECS suffers from the risk of data leakage. Existing secure CECS schemes are secure only if all edge servers are trusted. In other words, if any edge server is compromised, all cloud data (generated by IoT devices) will be leaked. Additionally, it is costly to request expected data from the cloud, which is linear with respect to the number of edge servers. To address the above problems, we propose a new secure data search and sharing scheme for CECS. Our scheme improves the existing secure CECS

scheme in the following two ways. First, it enables users to generate a publicand-private key pair and manage private keys by themselves. In contrast, the existing solution requires edge servers to manage users' private keys. Second, it uses searchable public-key encryption to achieve more secure, efficient, and flexible data searching. In terms of security, our scheme ensures the confidentiality of cloud data and secure data sharing and searching and avoids a single point of breakthrough. In terms of performance, the experimental results show that our scheme significantly reduces users' computing costs by delegating most of the cryptographic operations to edge servers. Especially, our scheme reduces the computing and communication overhead for generating a search trapdoor compared with the existing secure CECS scheme.

PAPER 8:

Title: A Preliminary Study on Data Security Technology in Big Data Cloud Computing Environment.

Authors: Zijiao Tang

Abstract: In the rapid development of Internet technology, the application of various new technologies has become more and more extensive. Among them, the development of big data technology and cloud computing technology is very rapid, and these technologies are of great help to improving the efficiency of data storage and management. However, the data system itself has certain data security problems in the big data cloud computing environment. In order to improve people's information security in the process of data processing and integration, we need to study data security protection technology in the big data cloud computing environment. Only in this way can we solve data and information security issues based on specific conditions and improve the reliability and security of data transmission.

PAPER 9:

Title: Challenges and security issues in cloud computing from two perspectives: Data security and privacy protection.

Authors: S. Mahdi Shariati; Abouzarjomehri; M. Hossein Ahmadzadegan

Abstract: Cloud computing is one of the most popular techniques in distributed computing which will increase scalability and flexibility in computer processing due to its ability to minimize the cost of time calculations. Cloud computing provides resources and shared services through the internet. Services will be delivered through the data center. Cloud computing provides an interesting business proposal for information technology industry, which without any additional investment, customers are able to do heavy processing by devices such as a mobile phone that has the resources including the web browser to run. On the other hand, cloud computing has been subject to many security issues. When a client delivers his data to a cloud provider for saving, there is the possibility of data loss. From the perspective of customers, cloud computing security concerns are still in place, in particular issues related to data security and privacy protection. In this paper, the challenges and security issues in cloud computing are investigated from two perspectives being data security and privacy protection.

PAPER 10:

Title: Block Design-Based Key Agreement for Group Data Sharing in Cloud Computing.

Authors: Jian Shen; Tianqi Zhou; Debiao He; Yuexin Zhang; Xingming Sun; Yang Xiang

Abstract: Data sharing in cloud computing enables multiple participants to freely share the group data, which improves the efficiency of work in cooperative environments and has widespread potential applications. However, how to ensure the security of data sharing within a group and how to efficiently share the outsourced data in a group manner are formidable challenges. Note that key

agreement protocols have played a very important role in secure and efficient group data sharing in cloud computing. In this paper, by taking advantage of the symmetric balanced incomplete block design (SBIBD), we present a novel block design-based key agreement protocol that supports multiple participants, which can flexibly extend the number of participants in a cloud environment according to the structure of the block design. Based on the proposed group data sharing model, we present general formulas for generating the common conference key K for multiple participants. Note that by benefiting from the (v,k+1,1) -block design, the computational complexity of the proposed protocol linearly increases with the number of participants and the communication complexity is greatly reduced. In addition, the fault tolerance property of our protocol enables the group data sharing in cloud computing to withstand different key attacks, which is similar to Yi's protocol.

PAPER 11:

Title: Mobile Cloud Computing Model and Big Data Analysis for Healthcare Applications.

Authors: Lo’ai A. Tawalbeh; Rashid Mehmood; Elhadj Benkhlifa; Houbing Song

Abstract: Mobile devices are increasingly becoming an indispensable part of people's daily life, facilitating to perform a variety of useful tasks. Mobile cloud computing integrates mobile and cloud computing to expand their capabilities and benefits and overcomes their limitations, such as limited memory, CPU power, and battery life. Big data analytics technologies enable extracting value from data having four Vs: volume, variety, velocity, and veracity. This paper discusses networked healthcare and the role of mobile cloud computing and big data analytics in its enablement. The motivation and development of networked healthcare applications and systems is presented along with the adoption of cloud computing in healthcare. A cloudlet-based mobile cloud-computing infrastructure

to be used for healthcare big data applications is described. The techniques, tools, and applications of big data analytics are reviewed. Conclusions are drawn concerning the design of networked healthcare systems using big data and mobile cloud-computing technologies. An outlook on networked healthcare is given.

PAPER 12:

Title: Detection of Data Leakage in Cloud Computing Environment.

Authors: Neeraj Kumar; Vijay Katta; Himanshu Mishra; Hitendra Garg

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.

CHAPTER 3
THEORETICAL BACKGROUND

CHAPTER 3

THEORETICAL BACKGROUND

3.1 IMPLEMENTATION ENVIRONMENT

Java Technology

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

The Java Programming Language

- The Java programming language is a high-level language that can be characterized by all of the following buzzwords:
    - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure
- With most programming languages, we either compile or interpret a program so that we can run it on our computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler,

first we 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.

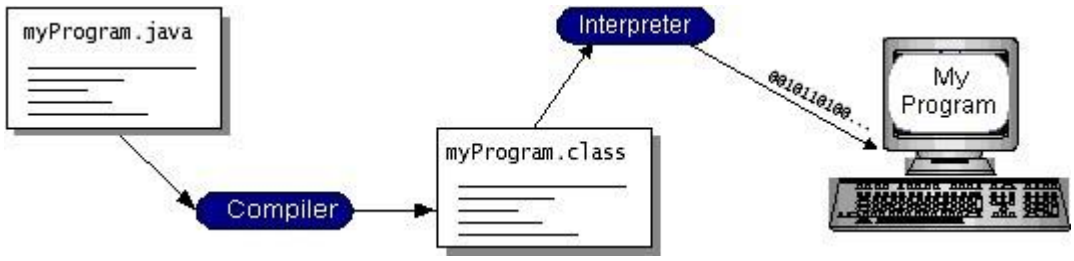


Figure 3.1.1 Java Programming

Java byte codes is 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. We can compile our 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.

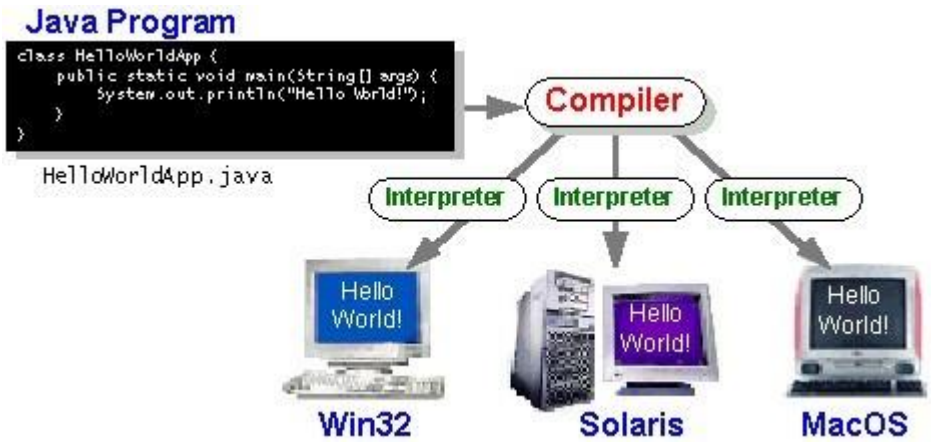


Figure 3.1.2 Java Compiler

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)*

Java VM is the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide. The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

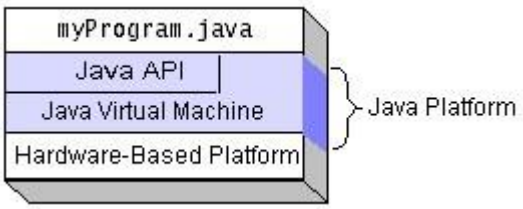


Figure 3.1.3 Java Platform

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, welltuned 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. 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, we can write many types of programs.

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

- The essentials: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- Applets: The set of conventions used by applets.
- Networking: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- Internationalization: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- Security: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- Software components: Known as 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.

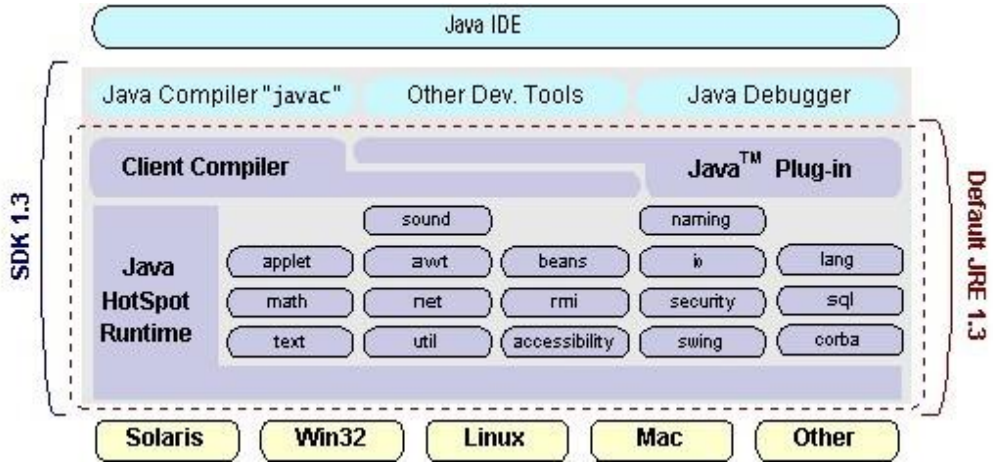


Figure 3.1.4 Java SDK Frame Work

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 to 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: The development time may be as much as twice as fast versus writing the same program in C++. Why? It write fewer lines of code and it is a simpler programming language than C++.

- Avoid platform dependencies with 100% Pure Java: It 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: It 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, 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 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 system by Windows 95. Rather, they are installed when it 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 an 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 , 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 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.

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 proceed the implementation using Java Networking.
And for dynamically updating the cache table we go for MS Access database.
Java ha two things: a programming language and a platform.
Java is a high-level programming language that is all of the following

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

Interpreted	Multithreaded
Robust Secure	Dynamic

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.

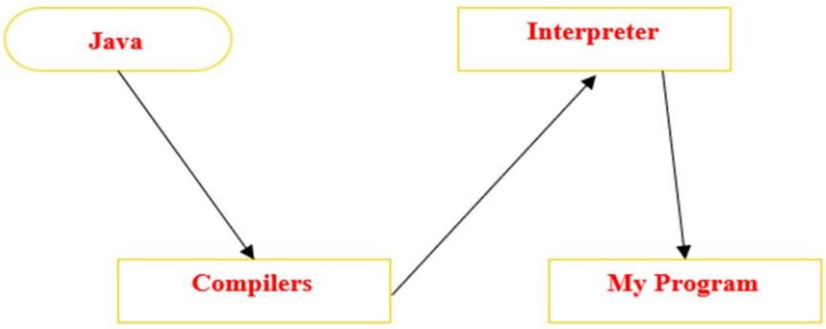


Figure 3.1.5 Java Interpreting

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

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

Networking

TCP/IP stack:

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

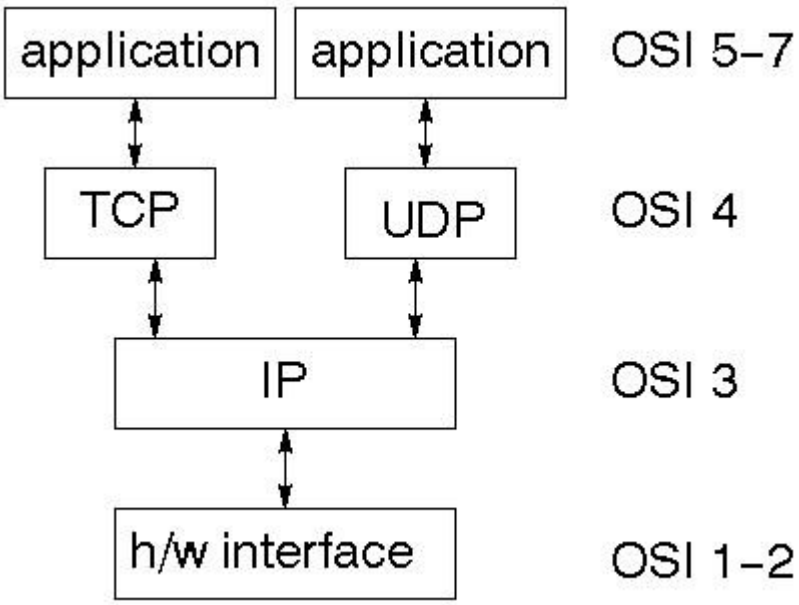


Figure 3.1.6 TCP/IP Stack

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

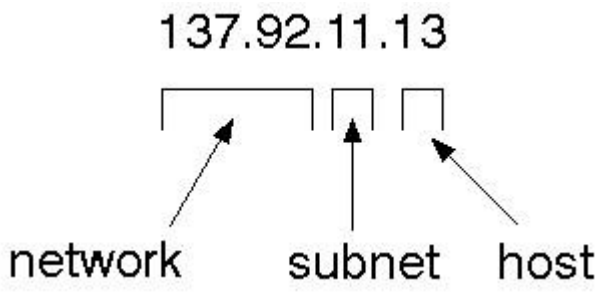


Figure 3.1.7 Total Address

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.

3.2 EXISTING SYSTEM

The existing system for data leakage detection in cloud computing environments employs a variety of tools and techniques aimed at identifying and preventing data breaches effectively. These include:

1. Encryption: Data encryption stands as one of the fundamental methods used to protect sensitive data in the cloud. By encoding data in such a way that it becomes unreadable without the corresponding decryption key, encryption ensures that even if unauthorized individuals gain access to the data, they cannot decipher its contents. Advanced encryption algorithms like AES (Advanced Encryption Standard) are commonly employed to secure data both during transmission and while at rest within cloud storage.

2. Access Control: Access control mechanisms play a crucial role in regulating and restricting access to data stored in the cloud. Technologies such as firewalls, which monitor and control incoming and outgoing network traffic, ensure that only authorized entities can communicate with cloud resources. Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) further enhance access control by actively monitoring network activities for suspicious behavior and taking appropriate action to mitigate potential threats.

3. Auditing and Monitoring: Regular auditing and monitoring of the cloud environment are essential components of a robust data leakage detection system. By continuously monitoring user activities, system events, and access logs, organizations can promptly detect any anomalous behavior or unauthorized access attempts. Auditing tools provide visibility into the entire data lifecycle, enabling administrators to track data movements, modifications, and access

patterns. Additionally, real-time alerts and notifications help security teams respond swiftly to potential security incidents.

4. Behavioral Analytics: Behavioral analytics tools leverage machine learning algorithms to establish baseline behavior patterns for users and devices accessing cloud resources. By continuously analyzing deviations from these norms, behavioral analytics solutions can identify potential insider threats, compromised accounts, or malicious activities that may lead to data leakage. This proactive approach enables organizations to detect and respond to security incidents before they escalate into full-fledged breaches.

By integrating these tools and techniques into their cloud computing environments, organizations can establish a multi-layered defense strategy against data leakage threats, thereby safeguarding sensitive information and maintaining regulatory compliance.

3.3 PROPOSED METHODOLOGY

3.3.1 SYSTEM ARCHITECTURE

Figure 3.2.1 System Architecture

The cloud server stores massive data on behalf of its clients (data owners). However, a malicious cloud server can delete some of the client's data (that are accessed infrequently) to save some space. Secure cloud storage protocols (twoparty protocols between the client and the server) provide a mechanism to detect if the server stores the client's data untampered. Based on the nature of the outsourced data, these protocols are classified as: secure cloud storage protocols for static data (SSCS) and for dynamic data (DSCS). For static data, the client cannot change her data after the initial outsourcing (e.g., backup/archival data). Dynamic data are more generic in that the client can modify her data as often as needed. In secure cloud storage protocols, the client can audit the outsourced data without accessing the whole data file, and still be able to detect unwanted changes in data done by a malicious server. During an audit, the client sends a random challenge to the server which produces proofs of storage (computed on the stored data) corresponding to that challenge. Secure cloud storage protocols are publicly verifiable if an audit can be performed by any third-party Auditor (TPA) using public parameters; or privately verifiable if an auditor needs some secret information of the client. The entities involved in a secure cloud storage protocol and the interaction.

3.3.2 DATA SET DESCRIPTION

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 clientside applications;

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

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

1. Map Visualizations

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

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

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

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

Time Series Chart Interactivity

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

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies,

thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

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

J2ME (Java 2 Micro edition):-

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

1. General J2ME architecture

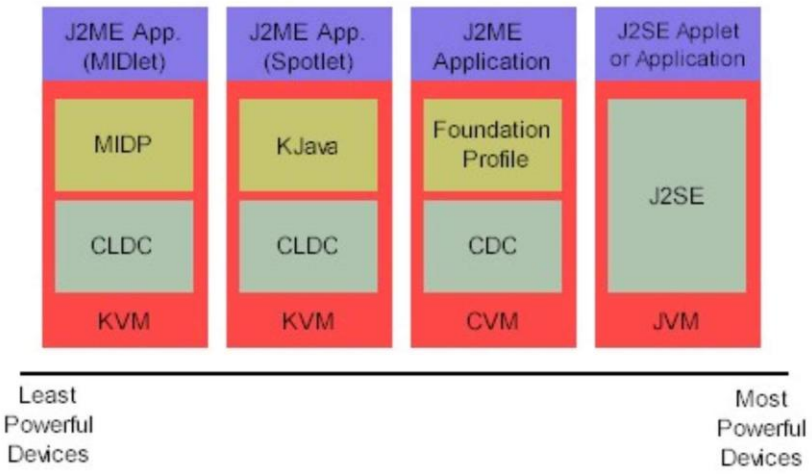


Figure 3.3.3.1 J2ME Architecture

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

2. Developing J2ME applications

when developing applications for smaller devices take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, explore packaging and deployment and the role preverification plays in this process.

3.Design considerations for small devices

Developing applications for small devices requires to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before begining 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. Managing the memory efficiently by setting object references to null on completion. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an asneeded 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?

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

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

Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However,

because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed

dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

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

* java.lang

* java.io

* java.util

* javax.microedition.io

* javax.microedition.lcdui

* javax.microedition.midlet

* javax.microedition.rms

3.3.3 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing 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 the privacy. Input Design considered the following things:

- What data should be given as input?

- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

3.3.3 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

3.3.5 MODULE DESIGN

Use Case Diagram

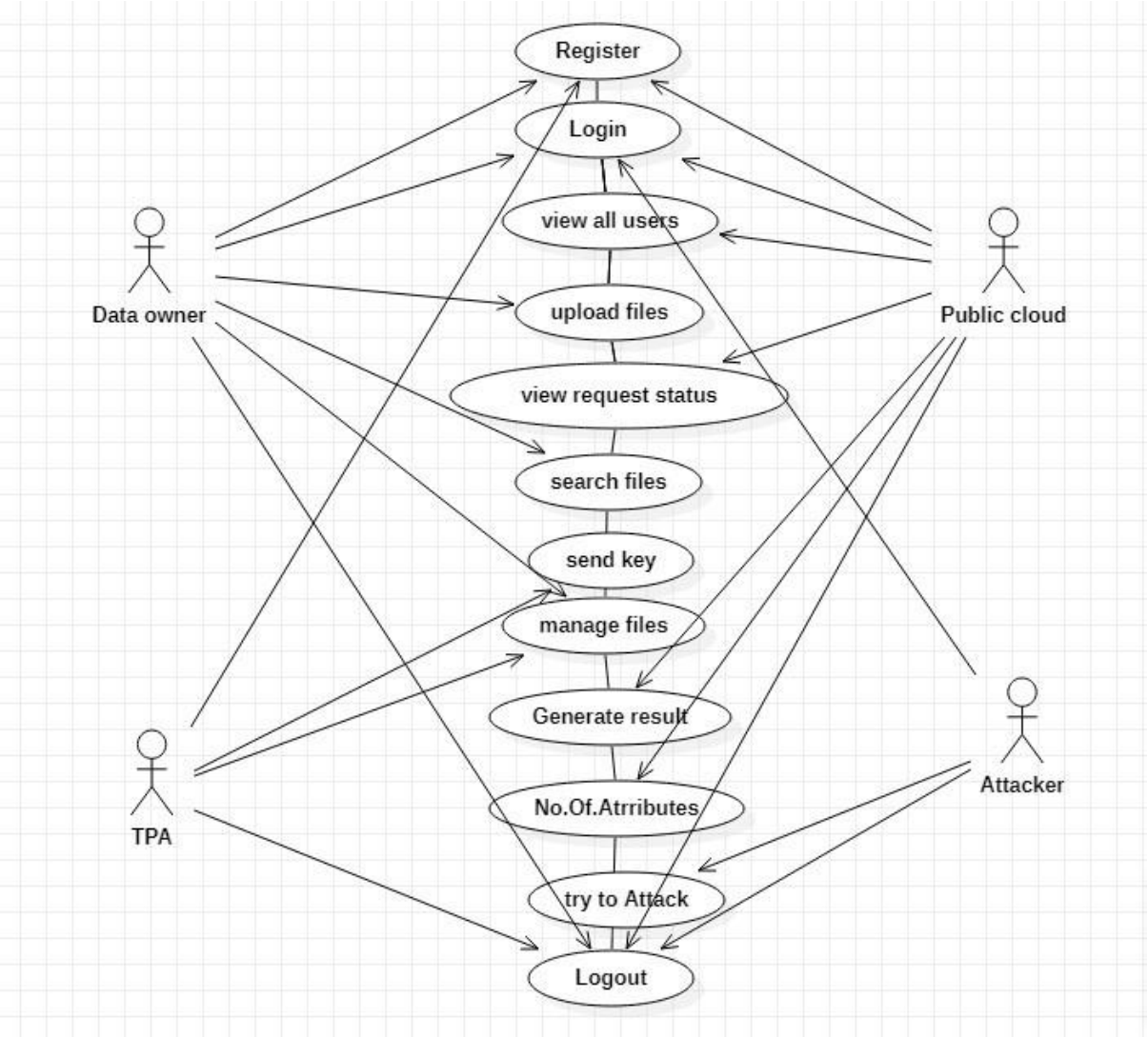


Figure 3.3.4.1 Use case diagram

Class Diagram

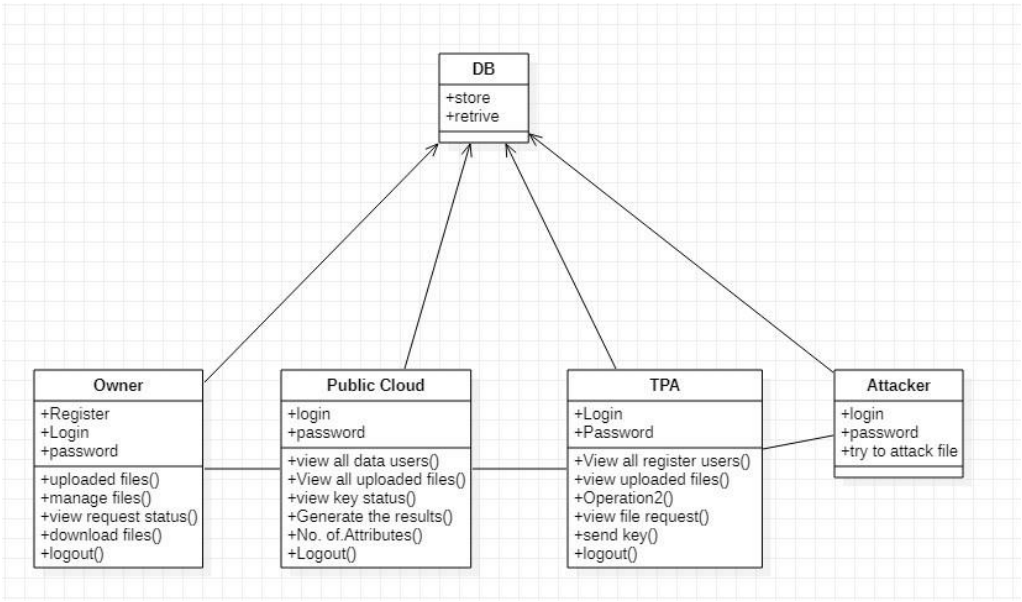


Figure 3.3.4.2 Class diagram

Sequence diagram

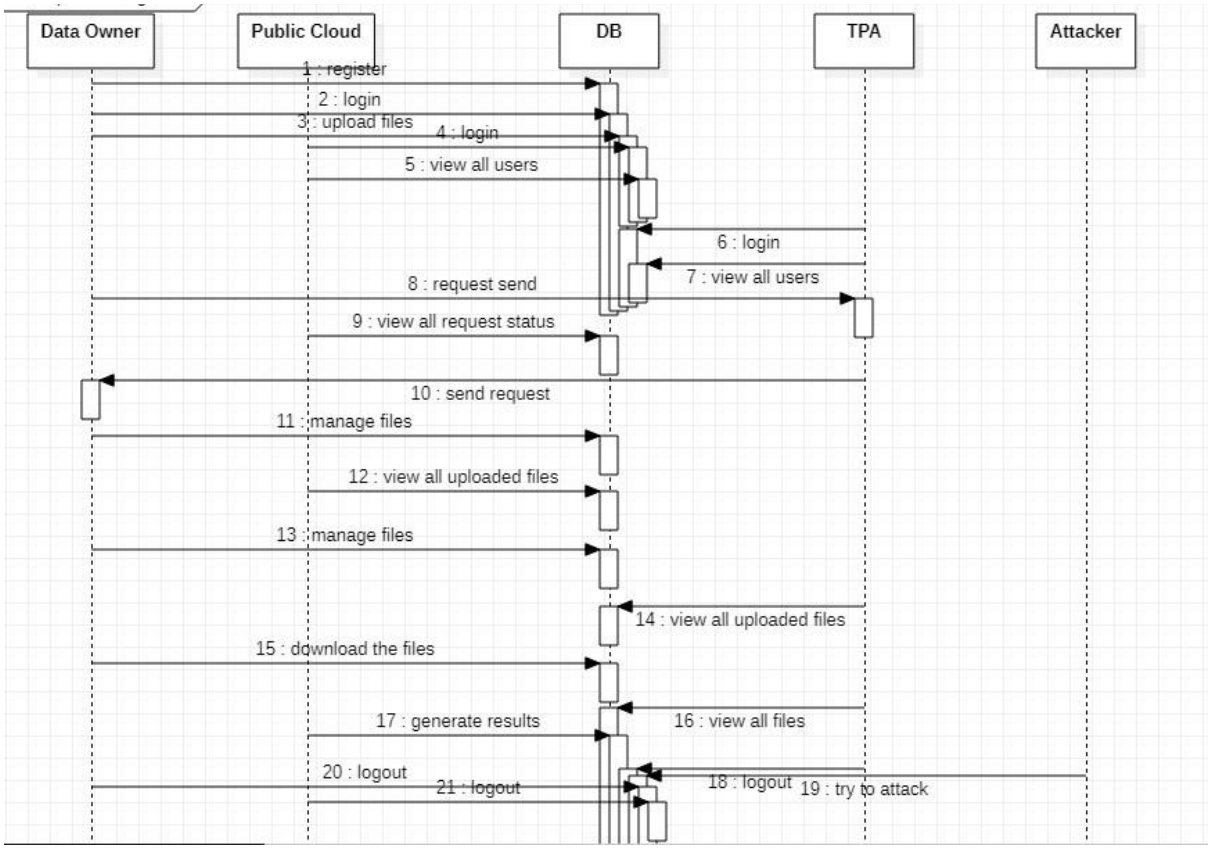


Figure 3.3.4.3 Sequence diagram

Deployment Diagram

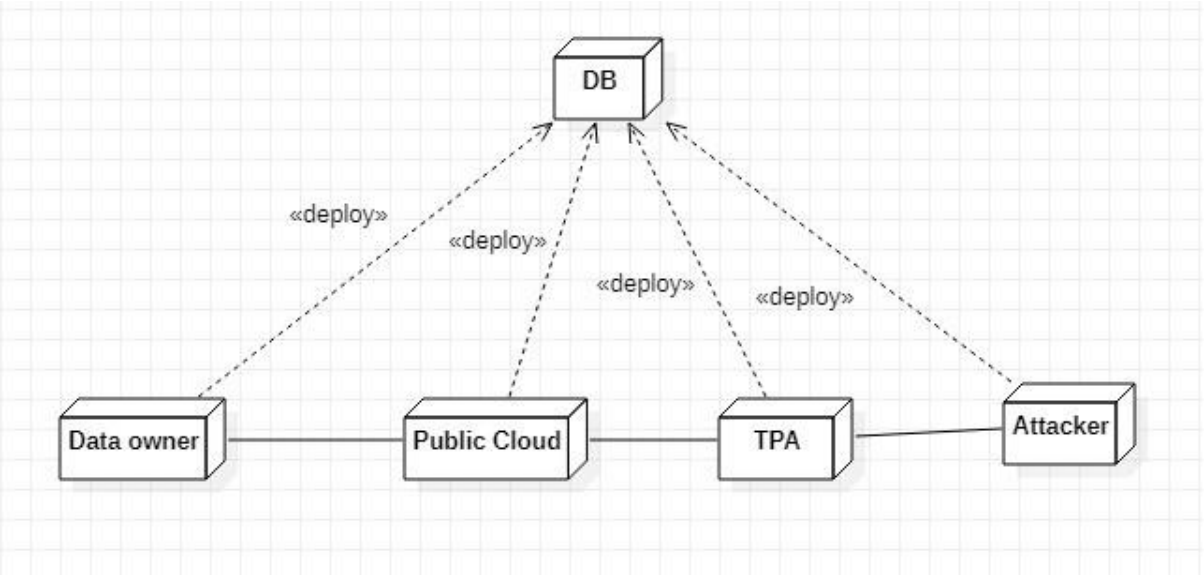


Figure 3.3.4.4 Deployment diagram

Dataflow Diagram

Level 0

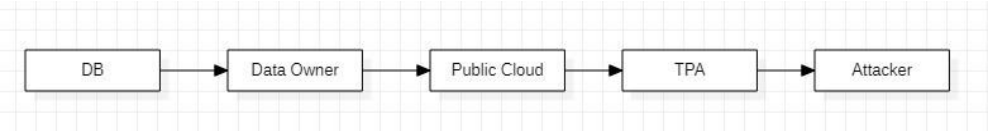


Figure 3.3.4.5 Level 0 diagram

Level 1

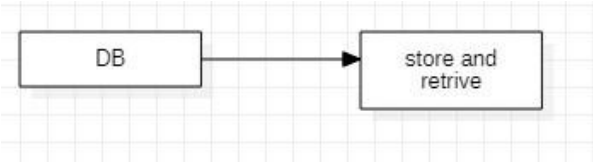


Figure 3.3.4.6 Level 1 diagram

Level 2

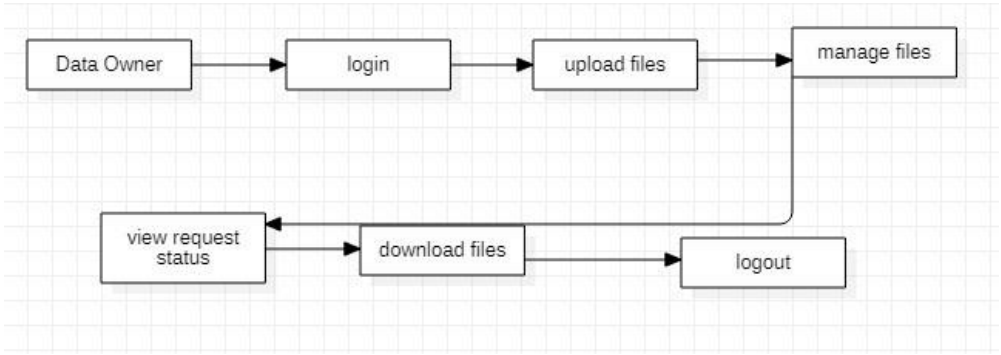


Figure 3.3.4.7 Level 2 diagram

Level 3

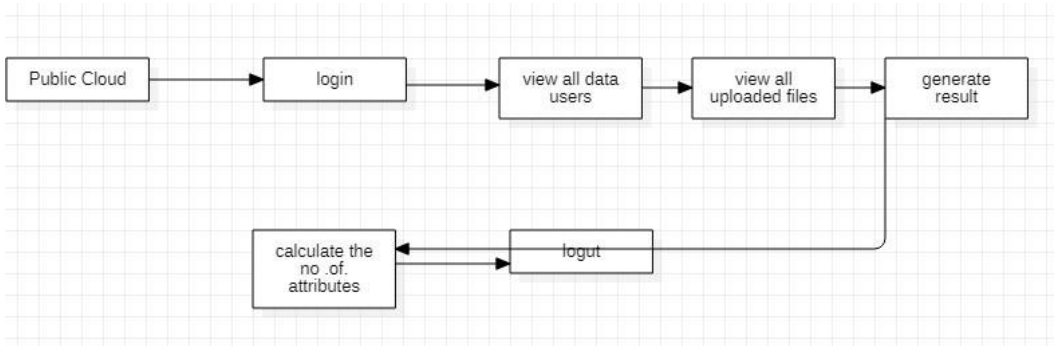


Figure 3.3.4.8 Level 3 diagram Level

Level 4

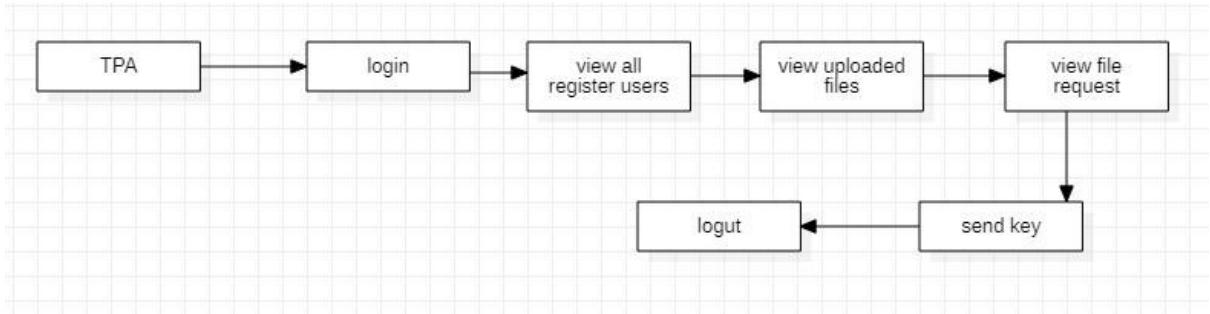


Figure 3.3.4.9 Level 4 diagram

Level 5

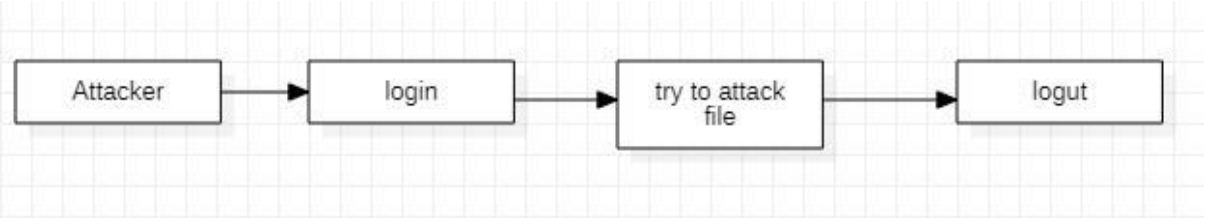
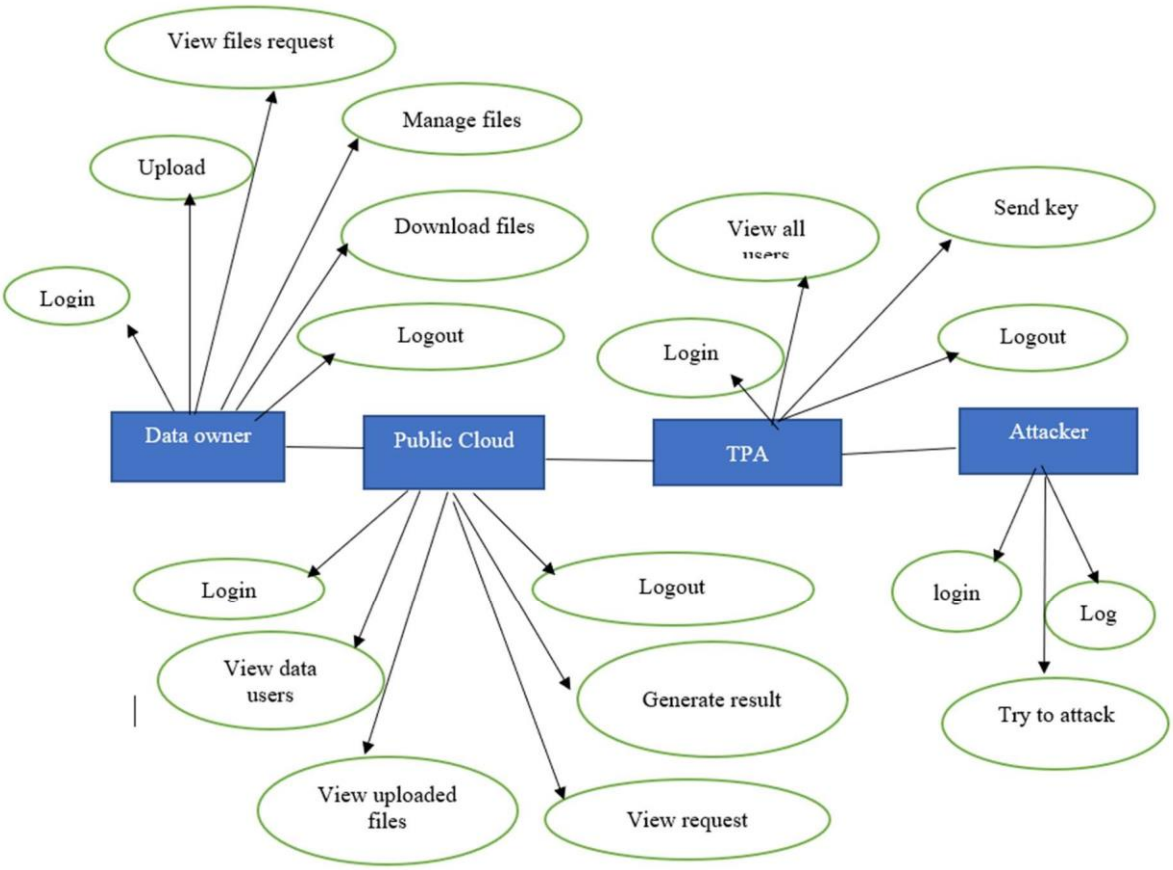


Figure 3.3.4.10 Level 5 diagram

ER Diagram



CHAPTER 4
SYSTEM IMPLEMENTATION

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 DESGIN PROCESS

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.

ADVANTAGES:

- 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.

4.2 ALGORITHMS

Collaborative filtering:

Collaborative filtering (CF) is a fundamental technique employed within recommender systems, enabling the provision of personalized recommendations tailored to individual user preferences and behaviors.

The mathematical expression can be expressed as

$$\begin{cases} 1 & \text{if Metric}(R, R\hat{}) > \text{Threshold} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

This method encompasses two distinct senses, each contributing to its effectiveness in delivering relevant suggestions:

1. Narrow Sense of Collaborative Filtering:

   In its narrow sense, collaborative filtering relies on the similarities between users and items to generate recommendations. By analyzing user-item interactions, such as ratings, purchases, or views, collaborative filtering algorithms discern patterns and preferences, facilitating the identification of items that align with a user's interests. This user-centric approach enables the system to recommend items that have been positively rated or favored by users with similar tastes and preferences.

2. Broad Sense of Collaborative Filtering:

   Collaborative filtering extends beyond the traditional user-item matrix, incorporating both user and item similarities to enhance recommendation accuracy. This broader approach allows collaborative filtering models to provide serendipitous recommendations, introducing users to novel content based on the preferences of similar users. For example, if user A exhibits similar preferences to user B, the system might recommend items that user B has enjoyed, thus facilitating the discovery of new and potentially appealing content.

Ciphertext:

Ciphertext is also known as encrypted or encoded information because it contains a form of the original plaintext that is unreadable by a human or computer without the proper cipher to decrypt it. Decryption, the inverse of encryption, is the process of turning ciphertext into readable plaintext.

4.3. MODULE DESCRIPTION

- In this project have 4 modules:

    1. Data Owner
    2. TA
    3. Cloud Server
    4. Attacker

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

TPA

- 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

PUBLIC CLOUD:

- 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

ATTACKER:

- Login
- Try to attack the file.

CHAPTER 5
RESULTS & DISCUSSION

5.1 TESTING PARAMETERS

Testcase ID	Input	Expected Output	Actual Output	Status
Testcase 001	User Register and Login	View Registration and login details	Registration and login page opened	Pass
Testcase 002	TPA(Third Party Administration)	View Tpa Login details	Tpa page opened and login details showed	Pass
Testcase 003	TPA Function	View all user and authorized user details	All users and authorized user detail page opened	Pass
Testcase 004	User details	Show User detail page	User detail page opened	Pass
Testcase 005	User detail page	Upload file to the user detail page	File uploaded successfully	Pass
Testcase 006	Managing files	View uploaded files	File storage page opened and showed	Pass
Testcase 007	Auditing action	Send request	Auditing request send to TPA successfully	Pass
Testcase 008	Auditing Check	View auditing files	Auditing files stored page opened	Pass
Testcase 009	Request for auditing	Sending request to admin	Request email send to admin successfully	Pass
Testcase 010	Attacker actions	Checking the file attacked	Email send to admin regarding attacked file	Pass

Table 5.1.1 Testcases

5.2 SYSTEM TESTING

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

TYPES OF TESTS

Unit testing

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

Integration testing

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

consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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

System Test

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

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at

least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

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

Unit Testing:

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

Test strategy and approach

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

Test objectives

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

Features to be tested

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

Integration Testing

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

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

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

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.

Action	Processing Time	Memory Usage	Detection Accuracy	False Positives	False Negatives	Throughput
User Register and Login	100 ms	20 MB	98%	2%	1%	10/s
TPA	120 ms	25 MB	95%	3%	2%	8/s
TPA Function	150 ms	30 MB	97%	1%	2%	6/s
User Details	80 ms	15 MB	99%	1%	0.5%	15/s
File Management	200 ms	35 MB	94%	4%	3%	5/s
Auditing Actions	180 ms	40 MB	96%	2.5%	1.5%	7/s

Request Handling	90 ms	18 MB	99%	0.5%	0.2%	12/s
Security Check	110 ms	22 MB	98.5%	1.5%	1%	9/s

Table 5.1.2 Performance parameters

CHAPTER 6
CONCLUSION & FUTURE WORK

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.

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.

APPENDICES

A.1 SDG GOALS

Sustainable Development Goal 9, "Industry, Innovation, and Infrastructure," is a pivotal component of the global agenda to address pressing challenges and foster sustainable development. At its core, SDG 9 aspires to build resilient infrastructure, stimulate inclusive and sustainable industrialization, and propel innovation. By strategically investing in transportation, energy, water, and sanitation infrastructure, countries can not only fortify their economic foundations but also uplift their citizens' quality of life. Encouraging the adoption of environmentally friendly technologies and sustainable industrial practices is integral to mitigating the environmental impact of industrialization. Moreover, the goal places a strong emphasis on fostering innovation, supporting research and development activities, and ensuring universal access to information and communication technology, with the aim of narrowing the digital divide and promoting global connectivity. SDG 9 also underscores the importance of financial inclusion, particularly for micro, small, and medium-sized enterprises, recognizing their role as engines of economic growth and job creation. Through international cooperation and partnerships, especially in providing financial, technological, and technical support to developing countries, SDG 9 envisions a world where resilient infrastructure, inclusive industrialization, and innovation collectively contribute to sustainable development. As we progress, ongoing efforts toward these targets will play a critical role in shaping a more equitable and resilient future for all.

A.2 CODING

```
/*
SQLyog Community v12.02 (32 bit)
MySQL - 5.5.29 : Database - intergity
*********************************************************************
*/


/*!40101 SET NAMES utf8 */;

/*!40101 SET SQL_MODE=''*/;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
CREATE DATABASE /*!32312 IF NOT EXISTS*/`intergity` /*!40100 DEFAULT CHARACTER SET latin1 */;

USE `intergity`;

/*Table structure for table `audit` */

DROP TABLE IF EXISTS `audit`;

CREATE TABLE `audit` (
  `id` varchar(200) DEFAULT NULL,
  `name` varchar(200) DEFAULT NULL,
  `fkey` varchar(200) DEFAULT NULL,
  `trap` varchar(200) DEFAULT NULL,
  `fname` varchar(200) DEFAULT NULL,
  `status` varchar(200) DEFAULT 'Waiting'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `audit` */

insert  into `audit`(`id`,`name`,`fkey`,`trap`,`fname`,`status`) values ('1','praba','6461','2ECB5F90560D91FA','READ ME.txt','Nodify send'),

/*Table structure for table `exe` */

DROP TABLE IF EXISTS `exe`;

CREATE TABLE `exe` (
  `id` int(20) DEFAULT NULL,
  `cname` varchar(200) DEFAULT NULL,
  `sqle` varchar(200) DEFAULT NULL,
  `status` varchar(200) DEFAULT 'Sql Injected',
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `exe` */

insert  into `exe`(`id`,`cname`,`sqle`,`status`,`date`) values (10,'Deepa','Select # from de','Sql Injected','2022-10-08 13:01:39'),(17,'Vel

/*Table structure for table `reg` */

DROP TABLE IF EXISTS `reg`;
```

```
CREATE TABLE `reg` (
  `sno` int(22) NOT NULL AUTO_INCREMENT,
  `uname` varchar(200) DEFAULT NULL,
  `pass` varchar(200) DEFAULT NULL,
  `email` varchar(200) DEFAULT NULL,
  `contact` varchar(200) DEFAULT NULL,
  `location` varchar(200) DEFAULT NULL,
  `status` varchar(222) DEFAULT 'Unauthorized',
  PRIMARY KEY (`sno`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `reg` */

/*Table structure for table `request` */

DROP TABLE IF EXISTS `request`;

CREATE TABLE `request` (
  `id` varchar(200) DEFAULT NULL,
  `name` varchar(200) DEFAULT NULL,
  `fkey` varchar(200) DEFAULT NULL,
  `trap` varchar(200) DEFAULT NULL,
  `fname` varchar(200) DEFAULT NULL,
  `fkkey` varchar(200) DEFAULT '-------',
  `status` varchar(200) DEFAULT 'Waiting'
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
insert  into `request`(`id`,`name`,`fkey`,`trap`,`fname`,`fkkey`,`status`) values ('1','praba','6461','2ECB5F90560D91FA','READ ME.txt','6461'

/*Table structure for table `revo` */

DROP TABLE IF EXISTS `revo`;

CREATE TABLE `revo` (
  `id` int(200) NOT NULL AUTO_INCREMENT,
  `uname` varchar(200) DEFAULT NULL,
  `count` int(200) DEFAULT '1',
  KEY `id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=latin1;

/*Data for the table `revo` */

insert  into `revo`(`id`,`uname`,`count`) values (1,'achu',2),(15,'Lakshmi',1);

/*Table structure for table `upload` */

DROP TABLE IF EXISTS `upload`;

CREATE TABLE `upload` (
  `reg` varchar(200) DEFAULT NULL,
  `name` varchar(200) DEFAULT NULL,
  `fkey` varchar(200) DEFAULT NULL,
  `dates` varchar(200) DEFAULT NULL,
  `trapdoor` varchar(200) DEFAULT NULL,
```

```
insert  into `upload`(`reg`,`name`,`fkey`,`dates`,`trapdoor`,`image`,`Attack`) values ('1','praba','6461','14/09/20 18:54:25','2ECB5F90560D91F

/*Table structure for table `ureg` */

DROP TABLE IF EXISTS `ureg`;

CREATE TABLE `ureg` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(200) DEFAULT NULL,
  `pass` varchar(200) DEFAULT NULL,
  `dob` varchar(200) DEFAULT NULL,
  `email` varchar(200) DEFAULT NULL,
  `cont` varchar(200) DEFAULT NULL,
  `address` varchar(200) DEFAULT NULL,
  `status` varchar(200) DEFAULT 'Unauthorized',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;

/*Data for the table `ureg` */

insert  into `ureg`(`id`,`name`,`pass`,`dob`,`email`,`cont`,`address`,`status`) values (1,'praba','praba','21/01/1997','cvsathyavani1999@gmail

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

USER LOGIN:

```
import java.io.IOException; import
java.io.PrintWriter; import
java.sql.Connection; import
java.sql.DriverManager; import
java.sql.PreparedStatement; import
java.sql.ResultSet; import
javax.servlet.RequestDispatcher; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 *
* @author ADMIN
 */
public class User_Login extends HttpServlet {

    /**
* Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
* methods.
```

```
    *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
    */
   protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
           throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
           PrintWriter out = response.getWriter();
HttpSession session=request.getSession();      try{
String name=request.getParameter("cname");
           String pass=request.getParameter("pass");

           Class.forName("com.mysql.jdbc.Driver");
           Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/intergity","root","ro
ot");
           PreparedStatement ps=con.prepareStatement("select * from ureg where
name='"+name+"' and pass='"+pass+"' and status='Authorized' ");
           System.out.println(ps);
           ResultSet                r=ps.executeQuery();
if(r.next()){
               String id=r.getString("id");
               String mail=r.getString("email");
              System.out.println(id+"                "+mail+"                "+name);
session.setAttribute("name",name);
               session.setAttribute("id", id);

               System.out.println("name"+name+"id"+id+"mail"+mail);
out.println("<script                        type=\"text/javascript\">");
out.println("alert(\"Welcome  "+name+"\")");
               out.println("</script>");
RequestDispatcher
rd=request.getRequestDispatcher("User_Home.jsp");
               rd.include(request, response);
           }
       }
       catch(Exception                        e){
System.out.println(e);
```

```
            out.println("<script type=\"text/javascript\">");
out.println("alert(\"Invalid Login\")");            out.println("</script>");
            RequestDispatcher
rd=request.getRequestDispatcher("loginregister.html");
rd.include(request, response);
        }
    }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on
the + sign on the left to edit the code.">
    /**
* Handles the HTTP <code>GET</code> method.
     *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
* Handles the HTTP <code>POST</code> method.
     *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
* Returns a short description of the servlet.
     *
```

```
* @return a String containing servlet description
     */
    @Override     public String
getServletInfo() {         return
"Short description";
    }// </editor-fold>

}
```

CLOUD LOGIN:

```
import java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.RequestDispatcher; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
* @author ADMIN
 */
public class Cloud_Login extends HttpServlet {

    /**
* Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
* methods.
     *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
     */
    protected          void          processRequest(HttpServletRequest          request,
HttpServletResponse response)
            throws              ServletException,              IOException              {
response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
```

```
            String name=request.getParameter("cname");
            String                                  pass=request.getParameter("pass");
System.out.println("username.."+name+" "+pass);
            if(name.equalsIgnoreCase("Cloud")&&pass.equalsIgnoreCase("Cloud")
){              out.println("<script type=\"text/javascript\">");
out.println("alert(\"Welcome Cloud Server\")");
                out.println("</script>");
RequestDispatcher
rd=request.getRequestDispatcher("Cloud_Home.jsp");
                rd.include(request, response);
            }           else{               out.println("<script
type=\"text/javascript\">");
out.println("alert(\"Invalid Login\")");
            out.println("</script>");
            RequestDispatcher         rd=request.getRequestDispatcher("Cloudl.jsp");
rd.include(request, response);
            }

        }

    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on
the + sign on the left to edit the code.">
    /**
* Handles the HTTP <code>GET</code> method.
     *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response)            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
* Handles the HTTP <code>POST</code> method.
     *
* @param request servlet request
* @param response servlet response
```

```
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
* Returns a short description of the servlet.
     *
* @return a String containing servlet description
     */
    @Override
    Public    String    getServletInfo()    {
return "Short description";
    }// </editor-fold>

}
```

ATTACKER:

```
import java.io.IOException; import
java.io.PrintWriter; import
javax.servlet.RequestDispatcher; import
javax.servlet.ServletException; import
javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Attacker extends HttpServlet {

    /**
* Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
* methods.
     *
   @throws IOException if an I/O error occurs
     */
```

```
   protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
           throws ServletException, IOException {
       response.setContentType("text/html;charset=UTF-8");
     PrintWriter out = response.getWriter();

           String name=request.getParameter("cname");
           String pass=request.getParameter("pass");
           System.out.println("username.."+name+" "+pass);
if(name.equalsIgnoreCase("Attacker")&&pass.equalsIgnoreCase("Attac
ker")){                out.println("<script type=\"text/javascript\">");
              out.println("alert(\"Welcome Attacker\")");
              out.println("</script>");
RequestDispatcher
rd=request.getRequestDispatcher("Attacker_Home.jsp");
              rd.include(request, response);
           }           else{                out.println("<script
type=\"text/javascript\">");
out.println("alert(\"Invalid Login\")");
out.println("</script>");
           RequestDispatcher       rd=request.getRequestDispatcher("Attacker.jsp");
rd.include(request, response);
           }

       }

   // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on
the + sign on the left to edit the code.">
   /**
* Handles the HTTP <code>GET</code> method.
    *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
* Handles the HTTP <code>POST</code> method.
    *
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
```

```
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
* Returns a short description of the servlet.
     *
* @return a String containing servlet description
     */
    @Override     public String
getServletInfo() {         return
"Short description";
    }// </editor-fold>

}
```

A.3 SCREEN SHOTS:

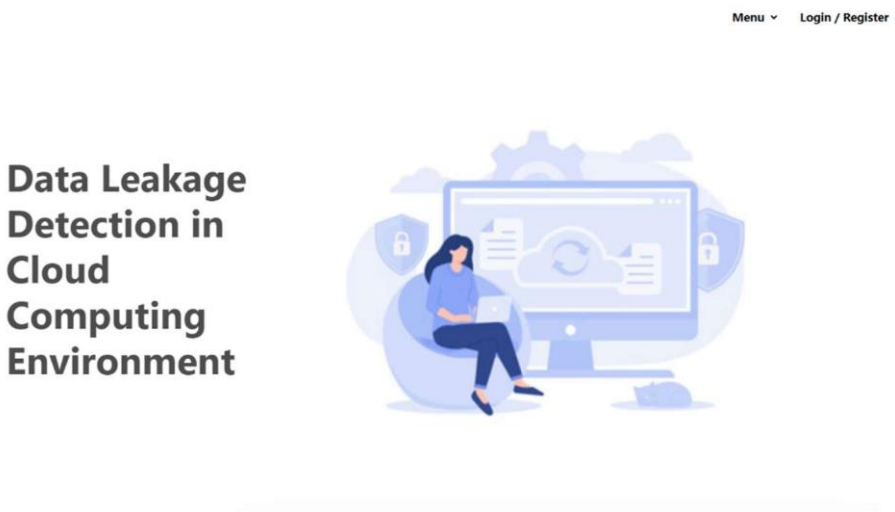


Figure A.1

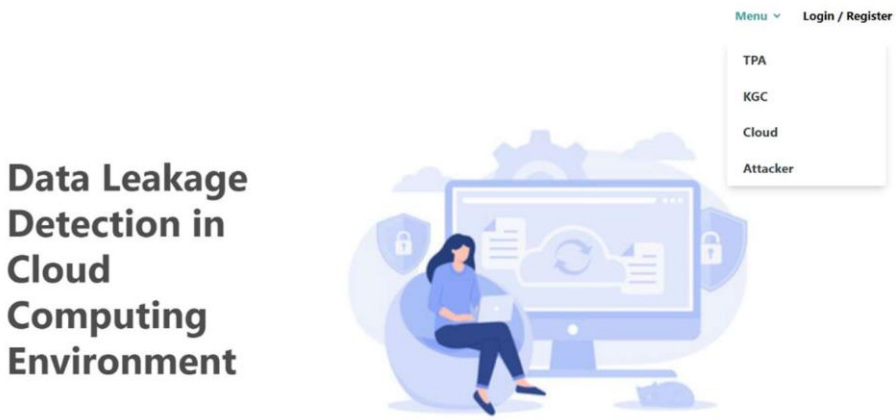


Figure A.2

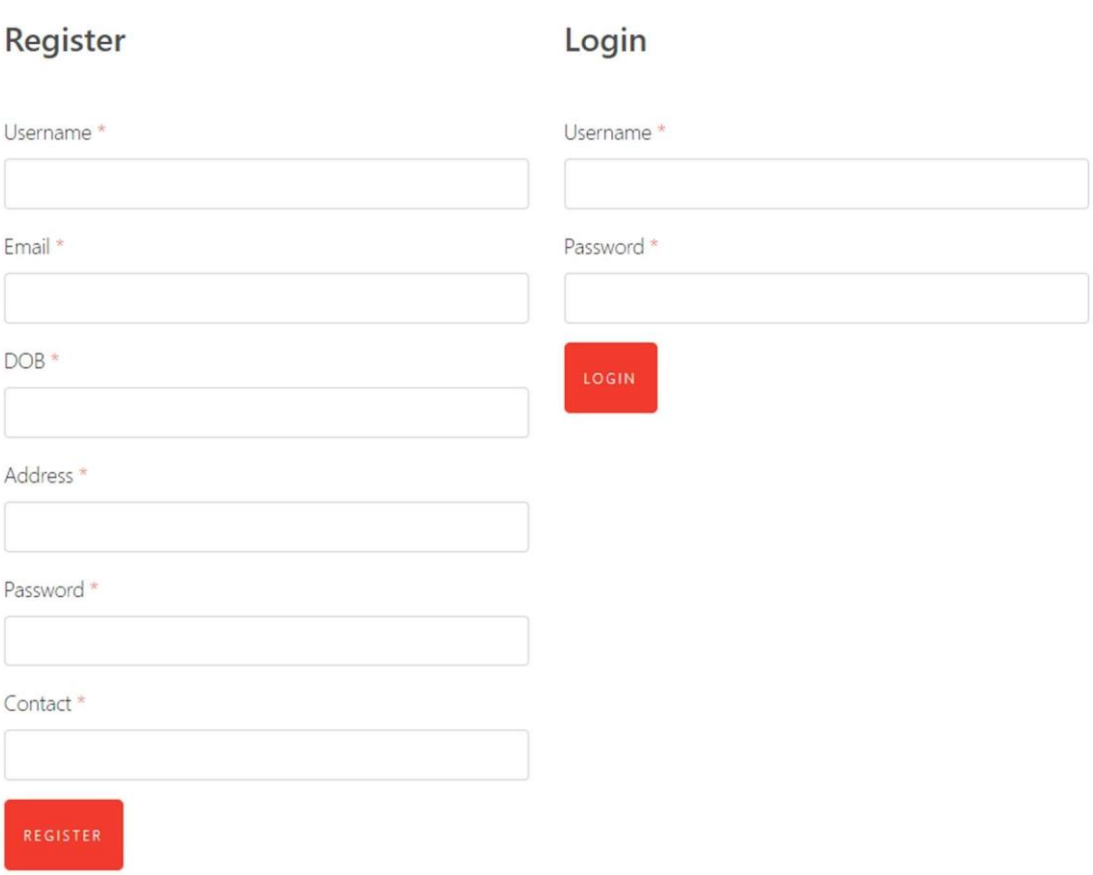


Figure A.3

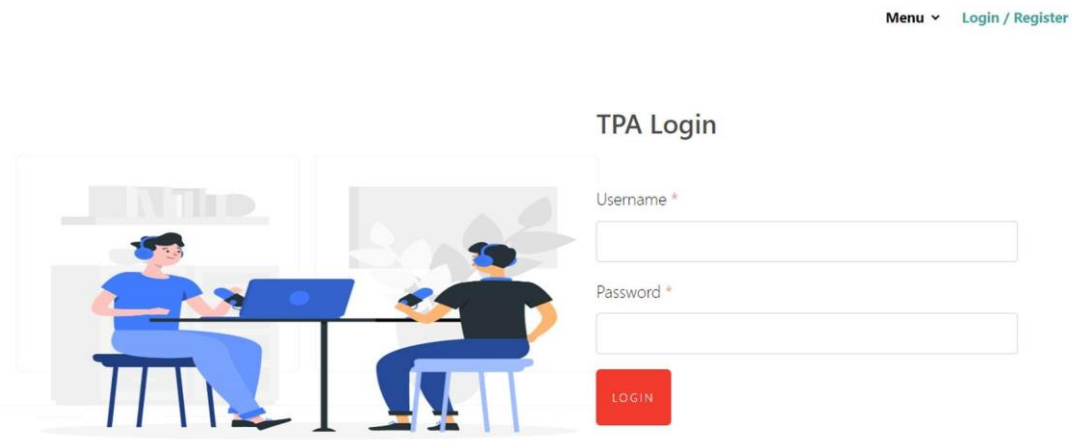


Figure A.4

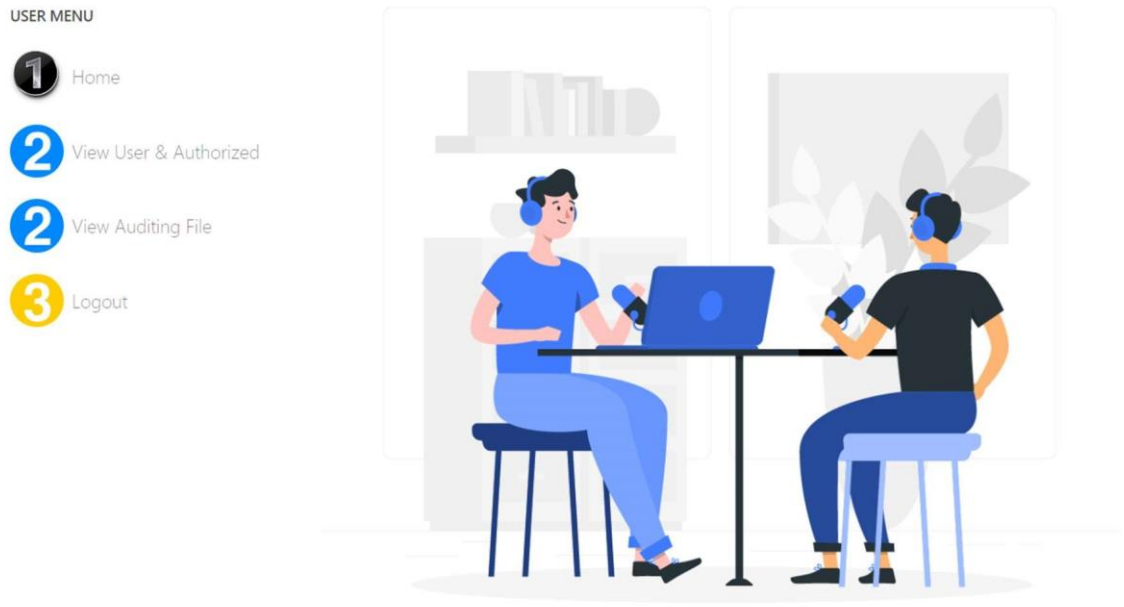


Figure A.5

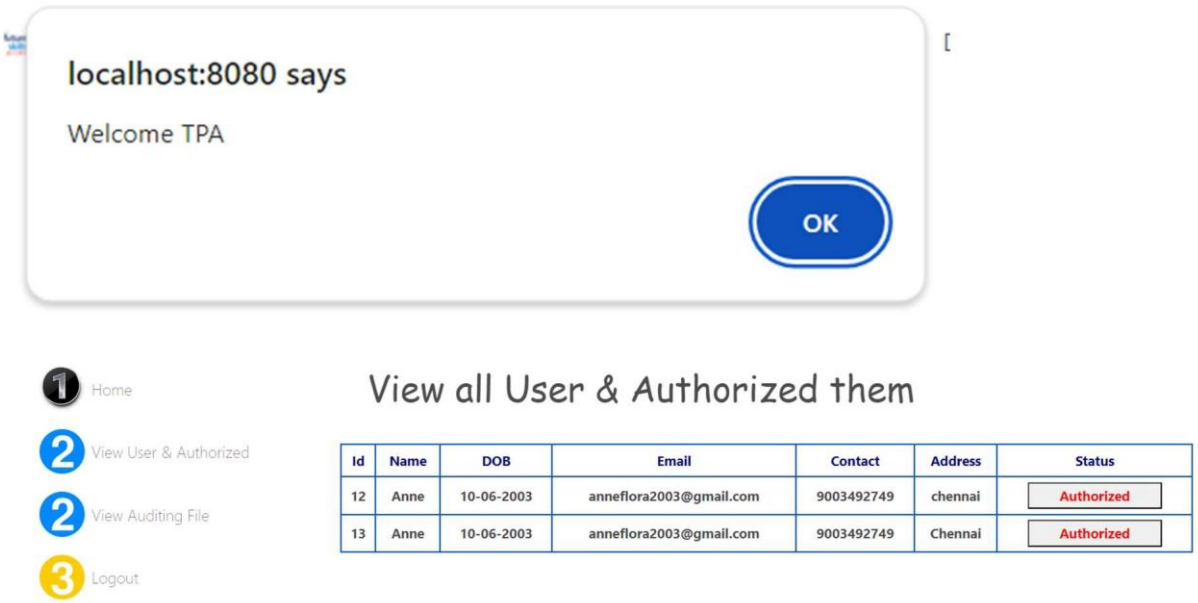


Figure A.6

Figure A.7

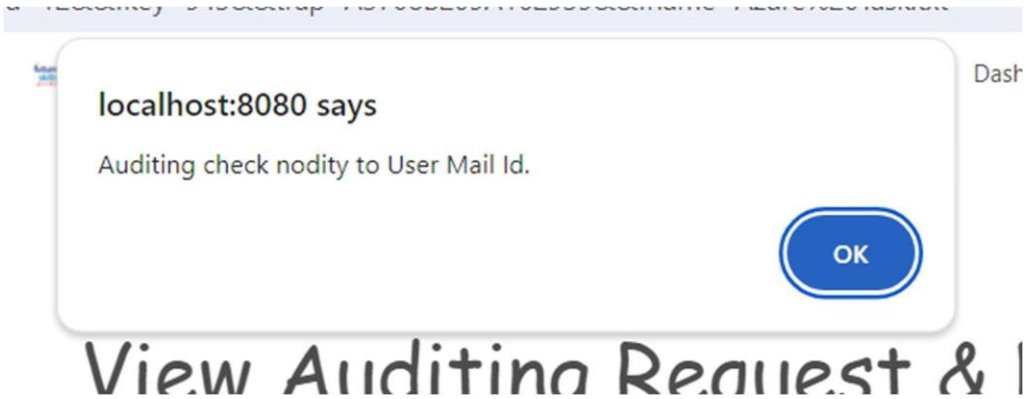


Figure A.8

Figure A.9

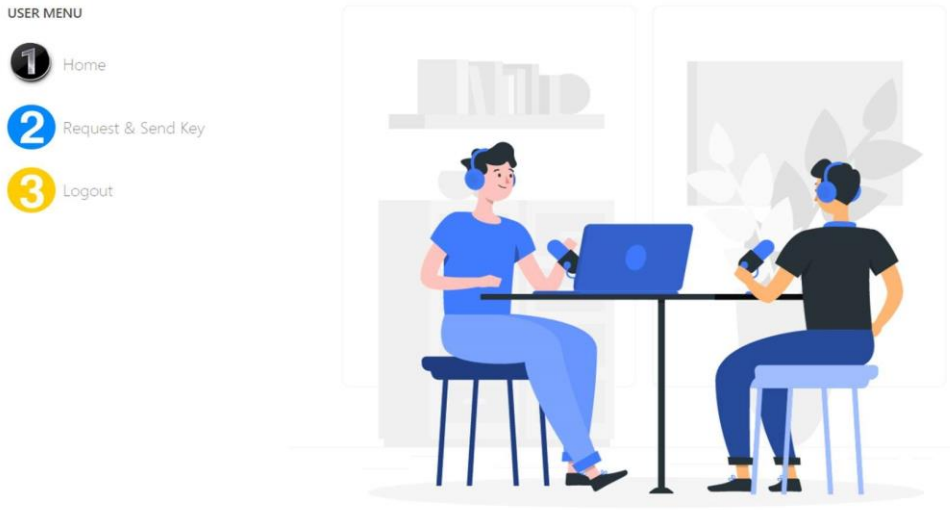


Figure A.10

Figure A.11

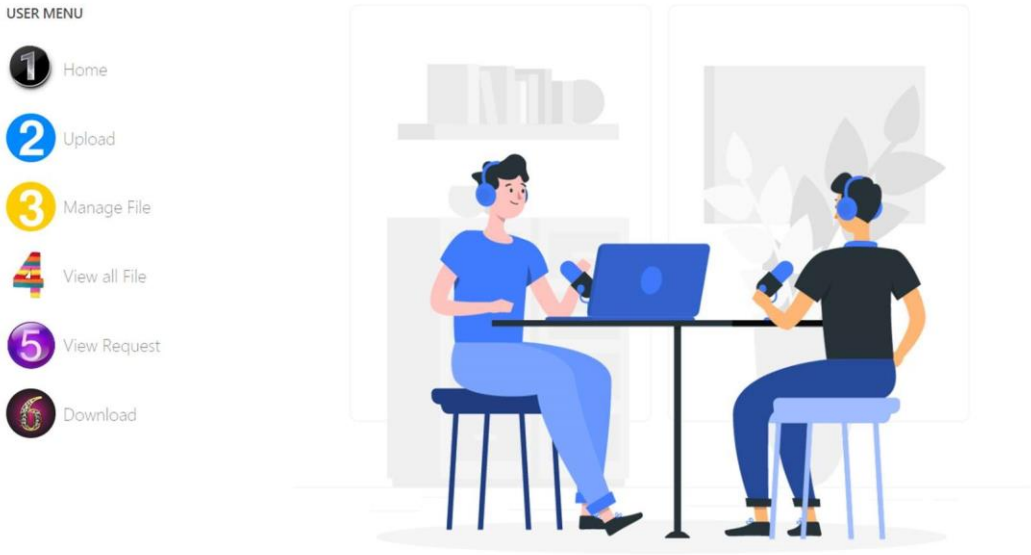


Figure A.12

Figure A.13

Figure A.14

Figure A.15

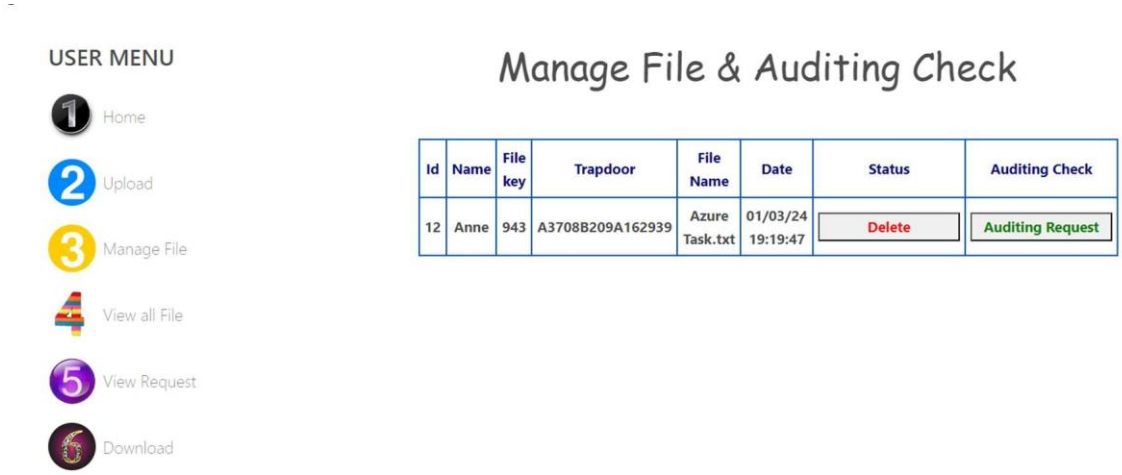


Figure A.16

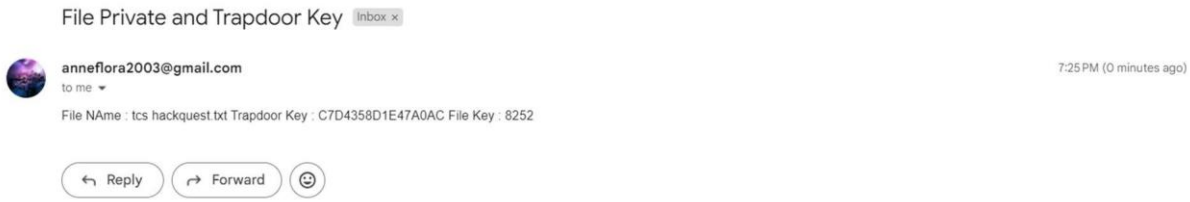


Figure A.17

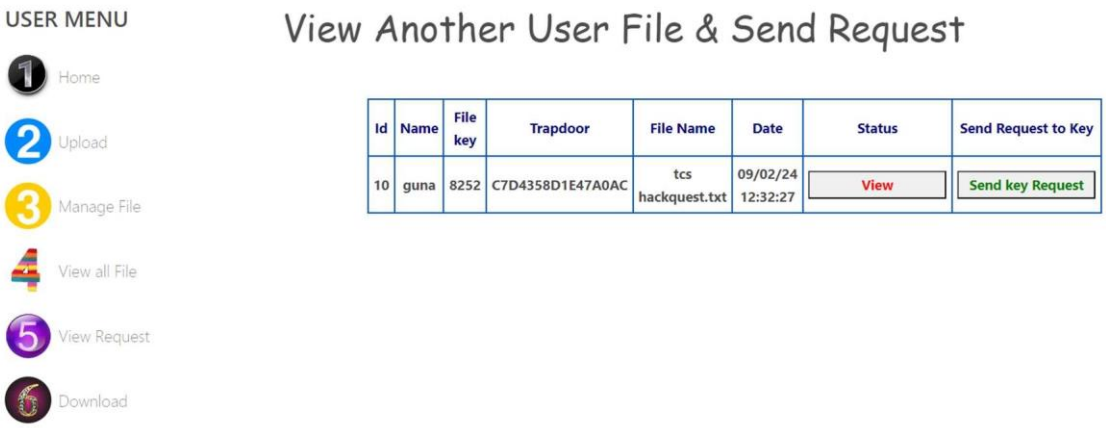


Figure A.18

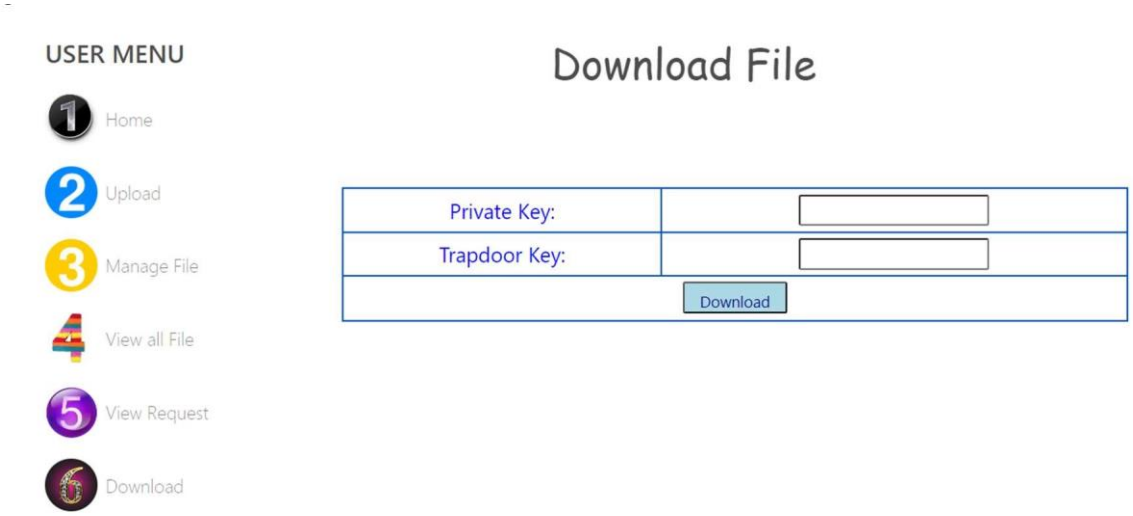


Figure A.19

A.4 Plagiarism Report

RE-2022-220762-plag-report

ORIGINALITY REPORT

8%	4%	2%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	2%
2	pace.ac.in Internet Source	1%
3	Kumar, Neeraj, Vijay Katta, Himanshu Mishra, and Hitendra Garg. "Detection of Data Leakage in Cloud Computing Environment", 2014 International Conference on Computational Intelligence and Communication Networks, 2014. Publication	1%
4	Submitted to Manipal University Student Paper	1%
5	www.buyya.com Internet Source	<1%
6	Submitted to Sreenidhi International School Student Paper	<1%
7	www.semanticscholar.org Internet Source	<1%

8	"Table of contents", IEEE Transactions on Information Forensics and Security, 2018 Publication	<1 %
9	Submitted to VIT University Student Paper	<1 %
10	Submitted to University of Bedfordshire Student Paper	<1 %
11	ieeexplore.ieee.org Internet Source	<1 %
12	O. Pandithurai, M. Poongodi, S. Pradeep Kumar, C. Gopala Krishnan. "A method to support multi-tenant as a service", 2011 Third International Conference on Advanced Computing, 2011 Publication	<1 %
13	Submitted to Asian Institute of Technology Student Paper	<1 %
14	www.researchgate.net Internet Source	<1 %
15	www.ir.juit.ac.in:8080 Internet Source	<1 %
16	K. Roslin Dayana, P. Shobha Rani. "Trust aware cryptographic role based access control scheme for secure cloud data storage", Automatika, 2023 Publication	<1 %

17	Submitted to University of East London Student Paper	<1 %
18	Submitted to Ohio University, Athens Student Paper	<1 %
19	veleco.cc Internet Source	<1 %
20	Ping Li, Tong Li, Heng Ye, Jin Li, Xiaofeng Chen, Yang Xiang. "Privacy-preserving machine learning with multiple data providers", Future Generation Computer Systems, 2018 Publication	<1 %
21	www.cloudbus.org Internet Source	<1 %
22	Ishu Gupta, Ashutosh Kumar Singh, Chung-Nan Lee, Rajkumar Buyya. "Secure Data Storage and Sharing Techniques for Data Protection in Cloud Environments: A Systematic Review, Analysis, and Future Directions", IEEE Access, 2022 Publication	<1 %
23	Leyou Zhang, Yilei Cui, Yi Mu. "Improving Security and Privacy Attribute Based Data Sharing in Cloud Computing", IEEE Systems Journal, 2020 Publication	<1 %

24	www.ajol.info Internet Source	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

ENHANCED DATA LEAKAGE DETECTION IN CLOUD COMPUTING ENVIRONMENT

ne Flora R
Departm of CSE
Chennai, India.

Annie S
Department of CSE
Chennai, India.

Sharnika S A
Department of CSE
Chennai, India.

Dr. T. Tamilvizhi, M.Tech., Ph.D
Associate professor
Department of CSE
Chennai, India.

Abstract-

In today's world, businesses rely heavily on internet technology to increase their efficiency by transferring data from one place to another. However, sharing important business information comes with its own set of risks, as any unethical employee could potentially disclose this information, leading to a data breach. To prevent such incidents, we provide examples of crime data during the search process, with the aim of identifying any employee leaking important information within the organization. Cloud computing services have become increasingly popular, and with more data being stored and processed in the cloud, it is essential to detect data leaks and unauthorized access to avoid data breaches. To this end, we propose to create a crime detection database suitable for cloud computing, with a method that involves combining static and dynamic analysis techniques to analyze and filter data. Static analysis of the source code is performed to identify potential sources of data exfiltration, while dynamic analysis is performed at runtime to verify data exfiltration. Additionally, machine learning algorithms are employed to improve the accuracy of crime reports by finding patterns and anomalies in data that may indicate the presence of data crime reports.

Keywords: cloud computing, data leakage, data security, encryption Archives, machine learning, untraceable.

I INTRODUCTION

Data breaches are a significant issue in today's business environment. This is because the data within an organization that is deemed confidential must be safeguarded from unwanted access. A data breach is defined as an unintended or deliberate disclosure of private information belonging to confidential organizations. It is crucial to protect this important information from unauthorized use.

Important documents include intellectual property documents, patent documents, technical documents, etc. takes place in modern day-to-day life. Many businesses already share importantcorpor ate information with stakeholders outside the organization's facilities. Therefore, the perpetrator who posted the item will not be identified. It is important to analyze data breaches when using the cloud to prevent such incidents. Due to the distributed nature of cloud computing (data is stored and processed in multiple locations), data breaches in the cloud environment can be difficult to detect. Additionally, the security technology used in non-cloud systems may not be sufficient to protect data from the cloud. What we recommend is to secure users' resources and detect people trying to access other people's information. Since one person can access other people's information through shared resources, we prevent this and ensure that the information is more secure by providing a private key for each file. In this way, anyone who tries to access or attack information will not be able to do so because administrators will supervise all work areas. By providing a different key, the message will be encrypted so that anyone who does not have the key cannot read the message unless they actually have the key, which is impossible.

II. LITERATURE SURVEY

Cloud computing provides a cost-effective and convenient way to share data. Transferring data to cloud servers raises concerns about its privacy. To safeguard sensitive information, many strategies are employed to improve access control for shared data.[1]"Improving security and privacy attribute based data sharing in cloud comput" by L. Zhang, Y. Cui, and Y. Mu describes Ciphertext-policy attribute-based encryption (CP-ABE) offers both convenience and security.Traditional CP-ABE prioritizes data confidentiality over user privacy, which is curr ly a significant concern. CP-ABE's disguised access policy assures data confidentiality and user privacy. This sugges approach provides selected security for the decisional n-BDHE problem and decisional linear assumption. The computational findings supp t the effectiveness, of the given approach.[3]" Secure fine-grained access control and data sharing for dynamic groups in the cloud" by S. Xu, G. ng, Y. Mu, and R. H. Deng, This paper proposes a secure and efficient access control and data sharing scheme for dynamic user groups. It involves (1) defining and enforcing access policies bas on data attributes,(2) allowing KGC to efficiently update user credentials,

and (3) allowing untrusted CSPs to perform expensive computation tasks without requiring a delegation key. We created an efficient revocable attribute-based encryption (RABE) scheme with ciphertext delegation by combining techniques from identity-based encryption (IBE), attribute-based encryption (ABE), subset-cover framework, and ciphertext encoding. Our cloud-based system provides granular access control and data sharing for on-demand services with dynamic user groups. Our proposed approach outperforms the current solution in terms of efficiency and scalability, as demonstrated by experimental results.[6]"A distribution model for data leakage prevention" by Y. Fan, Y. Rongwei, W. Lina, and M. Xiaoyan, This study presents a file distribution model that aims to prevent data leaking. This model selects the file allocation plan with the least overlap between user file sets, resulting in a high probability of identifying leakage sources. This allows for additional measures to improve information security. Simulation trials show that the approach accurately detects rogue users and the source of leaks.[15] "An efficient attribute-based encryption scheme with policy update and file update in cloud computing," by J. Li et al., This paper proposes an efficient ciphertext-policy ABE system that includes policy and file updates for cloud computing. First encryption generates ciphertext components that can be shared during policy and file updates. It reduces client storage and transmission expenses, as well as the PCSP's processing costs. The suggested technique is secure with the assumption of decision q-parallel bilinear Diffie-Hellman exponent (BDHE). Experimental simulations indicate that the suggested system is highly efficient for policy and file updates.[22] "Privacy-preserving machine learning with multiple data providers" by J. Li, T. Li, H. Ye, J. Li, X. Chen, and Y. Xiang This research proposes a novel technique for protecting data sets from multiple suppliers and the cloud. To protect different providers' privacy requirements, we employ public-key encryption with a double decryption method (DD-PKE) to encrypt the data sets with distinct publickeys. We use ϵ-differential privacy to protect the privacy of cloud-based data. Cloud servers, rather than data providers, add noise for ϵ-differential privacy in various data analytics. Our technique has been demonstrated to be secure in the security model. Our methodology outperforms traditional machine learning methods, as evidenced by the studies.

2.1 *EXISTING SYSTEM*

The current method for detecting data leakage in a cloud computing environment typically includes a variety of tools and strategies aimed at recognizing and preventing data breaches. This includes:

1. Encryption: Encrypting data is a popular way for cloud security. It entails encoding data in a way that makes it illegible without a key.
2. Access Control: Firewalls, IDS, and IPS prevent unauthorized access to cloud data.
3. Regular audits and monitoring of cloud environments helps avoid data leaks. It entails checking logs and other data sources for any unusual activity.
4. Data Loss Prevention (DLP)

Systems monitor and prevent data leaks in transit and at rest.

5. Two factor authentication:

Requires users to supply both a password and a security token before accessing data in the cloud.

6. Virtual Private Network (VPN): VPNs safe guard the connection between users' devices and the cloud, limiting illegal access.

III. PROPOSED SYSTEM

Our proposed solution for detecting data leakage in cloud computing environments combines machine learning algorithms with anomaly detection techniques. This system examines user activity and data access patterns to identify any irregularities that could suggest data leaking. The system may detect aberrant user behavior, such as accessing data during normal business hours or downloading huge amounts of data quickly. The technology can also monitor network traffic for anomalous data transfers or communications. To detect probable data leakage events, the system can utilize a combination of machine learning algorithms, including clustering and classification. The system can also use statistical analysis techniques, such regression and correlation analysis, to find trends and patterns in data.

3.1 *SOFTWARE ENVIRONMENT*

Java Technology

Java technology is a programming language and a platform that runs on top of other hardware-based platforms. The Java programming language is a high-level language used for the development of various applications. A platform refers to the hardware or software environment in which a program runs. The following are among the most popular operating systems and hardware combinations: Windows 2000, Linux, Solaris, and MacOS. Unlike most other platforms, the Java

platform is a software-only platform that operates 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)*

The Java program is insulated from hardware by the Java API and virtual machine. This is depicted in the figure.

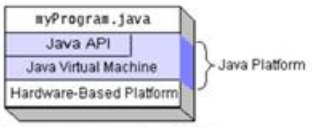


Fig.3.1 Architecture of Java Platform

ODBC

ODBC is a programming interface that allows app developers to connect with different database systems. Previously, developers had to use proprietary languages for each database which made coding difficult. However, ODBC has now made this choice almost irrelevant, allowing developers to focus on other aspects of their application.

JDBC

Sun Microsystems developed JDBC to create an independent database API for Java. JDBC provides a consis[24] interface to a wide range of RDBMSs through a gen[illegible]c SQL database access mechanism. This uniform interface is achieved through the use of "plug-in" database connectivity modules, or drivers. If a database vendor wants to have JDBC support, they must provide the driver for each platform on which the database and Java run. J2ME is highly secure because its programs are sandboxed, which means they cannot access other apps or data on the device. This helps prevent data leaks from rogue programs.

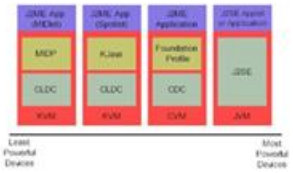


Fig.3.2 J2ME Architecture

J2ME is a technology that allows customization of the Java Runtime Environment (JRE) through the use of configurations and profiles. J2ME comprises a complete JRE, which includes a con[illegible]guration and a profile. The configuration specifies the JVM used, while the profile defines the application by incorporating domain-specific classes. The configuration creates the fundamental run-time environment, which consists of a set of core classes and a specific JVM that runs on specific types of devices.

3.2 ALGORITHM

Collaborative Filtering Algorithm is a technique used to identify abnormal patterns in us[illegible]em interactions in order to discover data leaks. In this method, user-item interactions are generally represented as a matrix where rows correspond to users, columns correspond to items, and each cell represents the user's interaction with an item (e.g., ratings, clicks, sales).

Normal Behavior Modeling:

Using past data, the collaborative filtering algorithm discovers typical user-item interaction patterns. It finds patterns and commonalities in user activity, like frequent interactions between users or individuals with similar tastes.

Anomaly Detection:

The system monitors user-item interactions to detect anomalies. It compares these interactions with the normal behavior model.

Metrics for Detection:

Various metrics are usually employed by anomaly detection algorithms to measure the departure of observed interactions from the expected normal behavior. These metrics could be statistical measures of deviance, similarity scores, or distance measurements.

Thresholding:

A threshold is applied to the detection metrics in order to ascertain whether an interaction is unusual.

The interaction is marked as possibly anomalous if the deviation is greater than the threshold.

Feedback Loop:

Over time, the model's performance can be enhanced by updating it with detected anomalies. The system can adjust to changing user behavior and enhance its accuracy in detecting anomalies by integrating feedback from discovered ones.

Mathematical Calculation

For mathematical calculation for the collaborative algorithm

Let us denote,

R as the matrix of user-item interaction that was identified

R^ as the user item interaction according to collaborativefiltering model.

Metric(R,R^) measures the difference between the interaction that were seen and the interaction that were expected. Threshold serves as the predetermined cutoff point for anomaly reporting.

The mathematical expression can be expressed as

1 if Metric(R,R^)>Threshold

0 otherwise (1)

IV. TEST CASE

Test case Identifier	Input To Testcase Scenario	Required Output	Received Output	Status
1.	User Register and Login	View Registration and login details	Registration and login page opened	Pass
2.	TPA(Third Party Administration)	View Tpa Login details	Tpa page opened and login details showed	Pass
3.	TPA Function	View all user and authorized user details	All users and authorized user detail page opened	Pass
4.	User details	Show User detail page	User detail page opened	Pass
5.	User detail page	Upload file to the user detail page	File uploaded successfully	Pass
6.	Managing files	View uploaded files	File storage page opened and showed	Pass
7.	Auditing action	Send request	Auditing request send to TPA successfully	Pass
8.	Auditing Check	View auditing files	Auditing files stored page opened	Pass
9.	Request for auditing	Sending request to admin	Request email send to admin successfully	Pass
10.	Attacker actions	Checking the file attacked	Email send to admin regarding attacked file	Pass

VI. FUTURE ENHANCEMENT

Data leakage in cloud computing is an active research topic that offers many interesting options for the future. Here are some possible suggestions: Develop powerful machine learning algorithms : Machine learning algorithms can be trained to detect unusual patterns in input and output data that would indicate a criminal record. However, existing algorithms may not be able to detect complex data breaches. Future research should prioritize developing more powerful machine learning systems to identify structural changes in infiltration data. In addition, more advanced personal protection systems should be developed. Technologies such as encryption, access control, and anonymity can be utilized to protect data in a cloud computing environment.

However these solutions also have disadvantages such as overwork and the possibility of insider attacks. In the future, efforts should be made to develop effective privacy protection strategies that can prevent data leaks and enhance data security.

In cloud computing, tracking data entry and transfer is difficult due to the vast amounts of data generated. Developing surveillance technologies to detect information breaches instantly is necessary. Future research should investigate new technologies like behavioral analysis and search engine optimization to improve crime reporting. Technological developments like Blockchain, edge computing, and IoT need to be assessed for their impact on investigating criminal records. New techniques need to be developed to tackle new problems that may arise.

VII. CONCLUSION

The proposed method will enhance security measures against data leakage problems. This technology enables real-time detection of data leakers and protects against both active and passive attacks. Moreover, this technique consumes less computational time and space.

As a result, this can be advantageous in a remote computing environment to prevent data leaking. Because the proposed technique is based on a symmetric algorithm, extending it to a web context with a large number of users accessing the data item is not viable. We can also use this strategy in asymmetric cryptography.

VIII. REFERENCES

[1] L. Zhang, Y. Cui, and Y. Mu, "Improving security and privacy attribute based data sharing in cloud computing " IEEE Syst. J., vol. 14, no. 1, pp. 387–397, Mar. 2020.

[2] Rupesh Mishra and DK Chitre. Data leakage and detection of guilty agent. International Journal of Scientific & Engineering Research, 3(6), 2012.

[3] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," IEEE Trans. Inf. Forensics Secure., vol. 13, no. 8, pp. 2101–2113, Aug. 2018 David Elliott Bell. Bell–la padula model. Encyclopedia of Cryptography and Security, pages 74–79, 2011. [4] Mukesh Singhal and Niranjan G Shivaratri. Advanced concepts in operating systems. McGraw-Hill, Inc., 1994. [5] 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.

[6] Y. Fan, Y. Rongwei, W. Lina, and M. Xiaoyan, "A distribution model for data leakage prevention," in Proc. Int. Conf. Mechatronic Sci., Elect. Eng. Comput., 2013, pp. 2617–2620.

[7]Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Performance comparison of the aes submissions, 1999.

[8] 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.

[9] 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.

[10] Hitendra GARG and Suneeta AGARWAL. A secure image based watermarking for 3d polygon mesh. SCIENCE AND TECHNOLOGY, 16(4): 287–303, 2013.

[11] Hitendra Garg and Suneeta Agrawal. Uniform repeated insertion of redundant water mark in 3d object. In Signal Processing and Integrated Networks (SPIN), 2014 International Conference on, pages 184–189. IEEE, 2014.

[12] CISSP Susan Hansche, CISSP John Berti, and Chris Hare. Official (ISC) 2 guide to the CISSP exam. CRC Press, 2003.
[13] D Elliott Bell and Leonard J La Padula. Secure computer system: Unified expo sition and multics interpretation. Technical rep ort, DTIC Document, 1976.
[14] David Elliott Bell. Looking back at the bell-la padula model. In ACSAC, volume 5, pages 337–351, 2005.
[15] J. Li et al., "An efficient attribute-based encryption scheme with policy update and file update in cloud computing," IEEE Trans. Ind. Inform., vol. 15, no. 12, pp. 6500–6509, Dec. 2019.
[16]Stallings William and William Stallings. C ryptography and Network Security, 4/E. Pears on Education India, 2006.
[17] Achal Kumar and Vibhav Prakash Singh. Digital watermarking using color image proce ssing using images
for transmitting secret information.
[18] JJK RUANAIDH and T PUN. Rotation, scale and translation invariant spread spectrum digital image water marking. Signal processing, 66(3):303–317, 1998.
[19] NIST FIPS Pub. 197. Announcing the Advanced Encryption Standard (AES), 20 01.
[20] AL Jeeva, Dr V Palanisamy, and K Kanagaram. Comparative analysis of perfor mance efficiency and security measures of some encryption algorithms. International Journal of Engineering Research and Applications(IJERA) ISSN, pages 2248–9622, 2012.
[21] E. Fujisaki, T. Okamoto, How to enhance the security of public-key encryption at minimum cost, in: International Workshop on Public Key Cryptography, 1999, pp. 53–68.
[22] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirectrevoc ation modes," in IMA, 2009, pp. 278–300.
[23] Y. Zheng, X. Yuan, X. Wang, J. Jiang, C. Wang, and X. Gui, "Toward encrypted cloud media center with secure deduplication, " IEEE Trans. Multimedia, vol. 19, no. 2, pp. 251–265, 2017.
[24] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in INFOCOM, 2010, pp. 534–542.
[25] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext
policy attributebased encryption," in
IEEE S&P, 2007, pp. 321–334.
[26] R. Li et al., "A lightweight secure data sharing scheme for mobile cloud computing," IEEE Trans. Cloud Comput., vol. 6, no. 2, pp. 344–357, Apr./Jun. 2018 [27] S. Wang
et al.,"An efficient file hierarchy attribute-based encryption scheme in cloud computing," IEEE Trans. Inf. Forensics Secur. , vol. 11, no. 6, pp. 1265–1277, Jun. 2016

REFERENECE

[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 Niranjan 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 AlNabhani, 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.