**ABSTRACT**

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

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

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

**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[1][2]. In the proposed work, our goal is to identify the guilty user when the organizational data have been leaked by some agent. In the proposed work, Bell-La Padula security model has been used which provide the analysis and design of secure computer systems. This model is called data confidentiality model. Bell-LaPadula model mainly focuses on data confidentiality issues and provides controlled access to classified information. In contrast to the Biba-Integrity model which describes rule for the protection of data integrity[3]. In this formal model, the entities in an information system are divided into subjects and objects. The notion of a ”secure state” is defined, and it is proven that each state transition preserves security by moving from one secure state to other secure state, thereby inductively proving that the system satisfies the security objectives of the model. The Bell-LaPadula model is built on the concept of a state machine with a set of allowable states in a computer system. A system state is defined to be secure if the only permitted access modes of subjects to objects are in accordance with a security policy. Cloud computing has become increasingly popular due to its ability to provide on-demand computing resources and cost-effective solutions. However, with the increasing usage of cloud computing services, the risk of data breaches and data leakage has also increased. Data leakage refers to the unauthorized disclosure of confidential or sensitive information, which can lead to significant financial and reputational damage for individuals and organizations.

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

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

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

**CLOUD COMPUTING**

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

**TYPES OF CLOUD COMPUTING**

Cloud computing is a broad term which refers to a collection of services that offer businesses a cost-effective solution to increase their IT capacity and functionality. There are three main types of cloud environment, also known as cloud deployment models. The models are.

* **Public cloud**

A public cloud is a type of computing in which a service provider makes resources available to the public via the internet. ... Some public cloud providers offer resources for free, while clients pay for other resources by subscription or a pay-per-usage model. Public clouds are the most common type of cloud computing deployment. Microsoft Azure is an example of a public cloud. In a public cloud, you share the same hardware, storage and network devices with other organisations or cloud “tenants,” and you access services and manage your account using a web browser.

* **Private cloud**

## The private cloud is defined as computing services offered either over the Internet or a private internal network and only to select users instead of the general public. Also called an internal or corporate cloud, private cloud computing gives businesses many of the benefits of a [public cloud](https://azure.microsoft.com/en-in/overview/what-is-a-public-cloud/) - including self-service, scalability and elasticity - with the additional control and customisation available from dedicated resources over a computing infrastructure hosted on-premises. In addition, private clouds deliver a higher level of security and privacy through both company firewalls and internal hosting to ensure operations and sensitive data are not accessible to third-party providers.

## Hybrid cloud

## Hybrid cloud is a solution that combines a private cloud with one or more public cloud services, with proprietary software enabling communication between each distinct service. A hybrid cloud strategy provides businesses with greater flexibility by moving workloads between cloud solutions as needs and costs fluctuate. Hybrid cloud refers to a mixed computing, storage, and services environment made up of on-premises infrastructure, private cloud services, and a public cloud—such as Amazon Web Services (AWS) or Microsoft Azure—with orchestration among the various platforms

## There are three main service models of cloud computing. They are

#### **IaaS (Infrastructure as Service)**

This is the most common service model of cloud computing as it offers the fundamental infrastructure of virtual servers, network, operating systems and data storage drives. It allows for the flexibility, reliability and scalability that many businesses seek with the cloud, and removes the need for hardware in the office. This makes it ideal for small and medium sized organisations looking for a cost-effective IT solution to support business growth. IaaS is a fully outsourced pay-for-use service and is available as a public, private or hybrid infrastructure.

#### **PaaS (Platform-as-a-Service)**

This is where cloud computing providers deploy the infrastructure and software framework, but businesses can develop and run their own applications. Web applications can be created quickly and easily via PaaS, and the service is flexible and robust enough to support them. PaaS solutions are scalable and ideal for business environments where multiple developers are working on a single project. It is also handy for situations where an existing data source (such as CRM tool) needs to be leveraged.

#### **SaaS (Software as a Service)**

This cloud computing solution involves the deployment of software over the internet to various businesses who pay via subscription or a pay-per-use model. It is a valuable tool for CRM and for applications that need a lot of web or mobile access – such as mobile sales management software. SaaS is managed from a central location so businesses don’t have to worry about maintaining it themselves, and is ideal for short-term projects.

**HOW CLOUD STORES THE DATAS?**

The storage outsourcing, 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 (two-party 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.

**NETWORK CODING TECHNIQUES**

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

Network coding techniques have been used to construct distributed storage systems where the client’s data are disseminated across multiple servers. However, they primarily aim to reduce the repair bandwidth when some of the servers fail. On the other hand, we explore whether we can exploit the algorithms involved in an SNC protocol to construct an efficient and secure cloud storage protocol for dynamic data (for a single storage server). Although dynamic data are generic in the sense that they support arbitrary update (insertion, deletion and modification) operations, append-only data (where new data corresponding to a data file are inserted only at the end of the file) find numerous applications as well. These applications primarily maintain archival as well as current data by appending the current data to the existing datasets. Examples of append-only data include data obtained from CCTV cameras, ledgers containing monetary transactions, medical history of patients, data stored at append-only databases, and so on.

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

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

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

**DISADVANTAGES**

* **C**annot store the large volume of data and computational is less.
* Malicious data can attack the client file.
* Less accuracy and the performance is low.

**2.2 PROPOSED SYSTEM**

The proposed system for “Data Leakage Detection Using Cloud Computing” consists of several components that work together to provide a comprehensive solution for data leakage detection and prevention in cloud computing environments.

1. *Data Classification:* The first component of the system involves classifying data based on its sensitivity level. The system uses machine learning algorithms to analyze data and classify it into different categories based on its sensitivity level. This component ensures that sensitive data is adequately protected and only accessed by authorized users.
2. *Access Control:* The second component of the system involves access control policies. Access control policies are applied to restrict user access to sensitive data. The system employs role-based access control policies to ensure that only authorized users can access sensitive data.
3. *Monitoring Mechanism:* The third component of the system involves monitoring user activities. The system uses machine learning algorithms to monitor user behavior and detect any suspicious activities that may indicate potential data leakage. The system can detect anomalies in user behavior and take appropriate actions to prevent data leakage.
4. *Cloud-Based Storage and Computing:* The fourth component of the system involves utilizing cloud-based storage and computing resources. The system uses cloud-based storage to store large volumes of data and computing resources to process data in real-time. This component provides scalability and flexibility to the system, ensuring that it can handle large volumes of data and adapt to changing user requirements.
5. *Reporting and Alerting:* The final component of the system involves reporting and alerting. The system generates reports on user activities and data access patterns, enabling administrators to monitor the system’s performance. The system also sends alerts to administrators when it detects potential data leakage, enabling them to take immediate action.

**SYSTEM REQUIREMENTS**

**3.1 HARDWARE REQUIREMENTS**

|  |  |
| --- | --- |
| System | Intel i3 and above |
| Hard Disk | 40GB |
| RAM | Minimum 4GB |
| Processor | 64-bit, four-core, 2.5 GHz minimum per core |

**3.2 SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| Front End  Language | HTML, CSS, JAVA, JSP  SERVELTS |
| Backend | My SQL |
| Operating System | Windows 10 or 11 |
| IDE | JAVADEVELIPEMENKIT |

**3.3 SOFTWARE DESCRIPTION**

**Software Environment**

**Java Technology**

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

**The Java Programming Language**

* The Java programming language is a high-level language that can be characterized by all of the following buzzwords:
* Simple
* Architecture neutral
* Object oriented
* Portable
* Distributed
* High performance
* Interpreted
* Multithreaded
* Robust
* Dynamic
* Secure
* With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer.
* You can think of Java byte codes as the machine code instructions for the JavaVirtual Machine(Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The by code scan 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 Solar is workstation, or on an iMac.





### The Java Platform

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

The Java platform has two components:

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

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

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



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

## What Can Java Technology Do?

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

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

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

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

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



## How Will Java Technology Change My Life?

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

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

### ODBC

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

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

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

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

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

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

**JDBC**

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

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

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

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

### JDBC Goals

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

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

1. **SQL Level API**

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

1. **SQL Conformance**

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query

statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

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

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

1. **Keep it simple**

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

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

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

1. **Keep the common cases simple**

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

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

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

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

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

Simple Architecture-neutral

Object-oriented Portable

Distributed High-performance

Interpreted multithreaded

Robust Dynamic

Secure

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

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

**JavaProgram**

**Compilers**

**Interpreter**

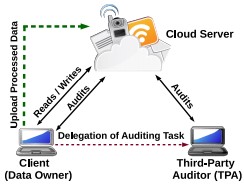
**My Program**

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

**SYSTEM DESIGN**

**4.1 SYSTEM ARCHITECTURE**

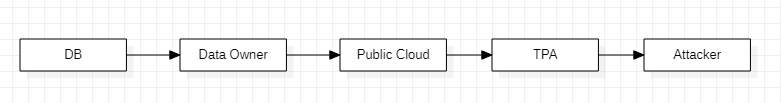
SYSTEM ARCHITECTURE



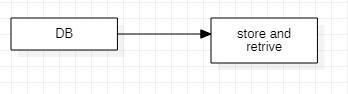
The cloud server stores massive data on behalf of its clients (data owners). However, a malicious cloud server can delete some of the client’s data (that are accessed infrequently) to save some space. Secure cloud storage protocols (two-party protocols between the client and the server) provide a mechanism 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.

**4.2 DATA FLOW DIAGRAM**

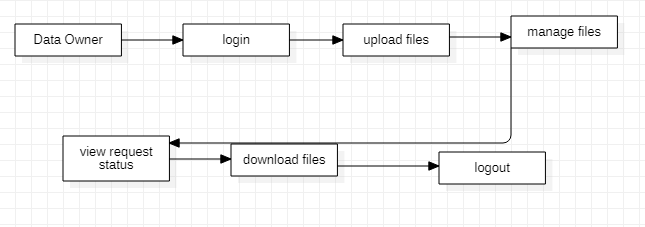
**Level 0**



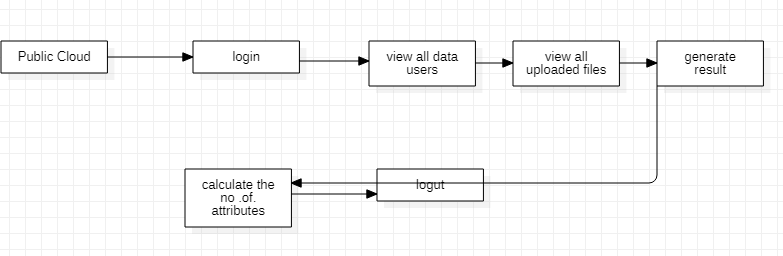
**Level 1**



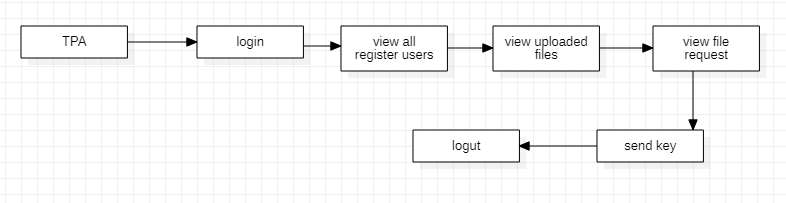
**Level 2**



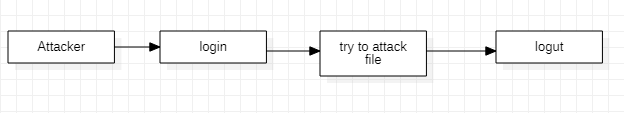
**Level 3**



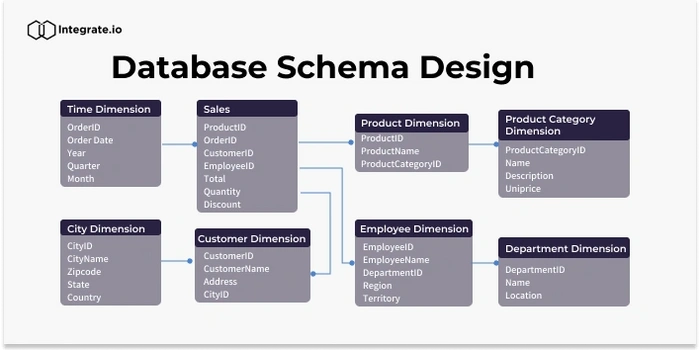
**Level 4**



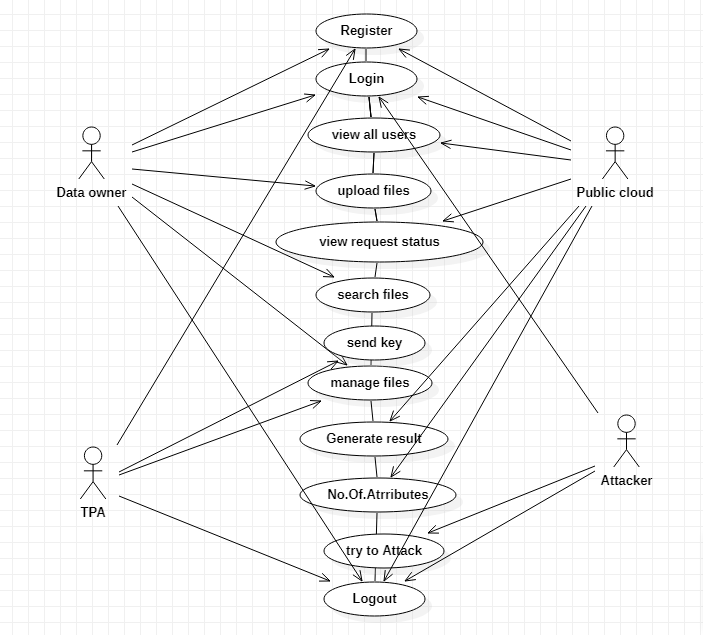
**Level 5**



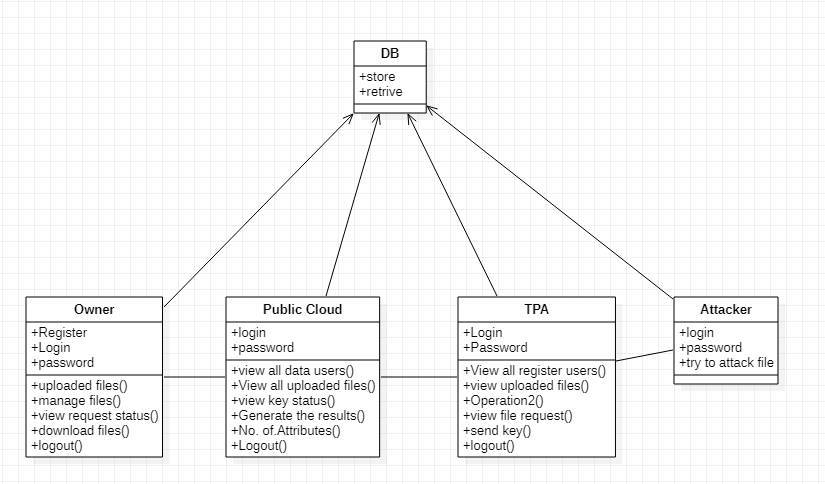
**4.3 DATABASE DESIGN**

****

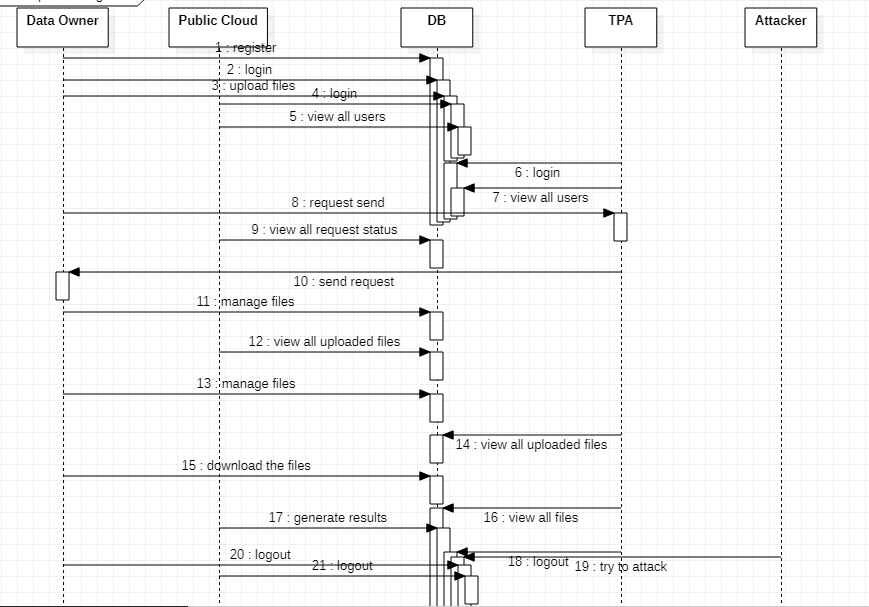
**4.4 UML DIAGRAMS**



**Class Diagram**



**Sequence diagram**



**4.5 ER DIAGRAM**

Data owner

Attacker

TPA

Public Cloud

### SYSTEM TESTING

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

**TYPES OF TESTS**

**Unit testing**

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

Unit

tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

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

**Functional test**

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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

**System Test**

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

**White Box Testing**

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

**Black Box Testing**

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

**6.1 Unit Testing:**

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

**Test strategy and approach**

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

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

**Features to be tested**

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

# 6.2 Integration Testing

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

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

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

**6.3 Acceptance Testing**

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

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

**SYSTEM IMPLEMENTATION**

**6.1 SYSTEM DESCRIPTION**

The implementation of the proposed data leakage detection system in cloud computing environments involves the integration of several key components, including QR code embedding, AES encryption, and PDF manipulation. Below, we outline the implementation details of each component:

### QR Code Embedding:

Utilized a Java library to read QR code images and convert them into Base64 encoded strings. Leveraged the Apache PDFBox library to access the metadata of PDF documents and embed the QR code data seamlessly. Implemented a function to concatenate the QR code data with existing metadata or create a new metadata field if necessary. Ensured that the embedding process maintains the integrity and confidentiality of the document's content, making the QR code imperceptible to the naked eye.

### AES Encryption:

Employed the AES encryption algorithm to encrypt sensitive data stored in the cloud. Integrated AES encryption into the data storage and retrieval processes to ensure that data remains encrypted at rest. Generated and managed encryption keys securely to prevent unauthorized access to the encrypted data.

### PDF Manipulation:

Used the Apache PDFBox library to load, modify, and save PDF documents. Developed functions to extract metadata from PDF documents and update metadata fields with the embedded QR code data. Implemented error handling to handle exceptions during PDF manipulation operations, such as file loading/saving errors or metadata extraction failures. Ensured compatibility with various PDF document formats and versions to support a wide range of use cases and scenarios.

### Integration and Testing:

Integrated the QR code embedding, AES encryption, and PDF manipulation components into a cohesive system. Conducted extensive testing to validate the functionality, performance, and security of the implemented system. Tested the system with various PDF documents, QR code images, and data encryption scenarios to ensure robustness and reliability. Implemented logging and monitoring mechanisms to track system activities and detect anomalies or errors during operation.

Overall, the implementation of the data leakage detection system involved the integration of QR code embedding, AES encryption, and PDF manipulation functionalities into a robust and reliable system. Through meticulous development, testing, the implemented system offers comprehensive data protection and leakage detection capabilities in cloud computing environments.

* 1. **MODULE DESCRIPTION**

This module provides an in-depth understanding of data leakage detection techniques within the context of cloud computing environments. As organizations increasingly adopt cloud services for data storage and processing, the risk of data **leakage becomes a** significant concern. Students will explore various strategies**,** algorithms, and tools used to detect and prevent data leakage incidents in cloud environments. The module covers theoretical concepts as well as practical implementation aspects to equip students with the necessary skills to identify, mitigate, and respond to data leakage threats effectively.

Module Objectives:

1. Understand the concept of data leakage and its implications in cloud computing environments.

2. Explore different types of data leakage threats and vulnerabilities specific to cloud platforms.

3. Study traditional and emerging data leakage detection techniques and algorithms.

4. Learn about the role of encryption, access controls, and authentication mechanisms in data leakage prevention.

5. Gain practical experience in implementing data leakage detection solutions using relevant tools and technologies.

6. Analyze real-world case studies and scenarios to assess the effectiveness of data leakage detection methods.

7. Develop skills in formulating strategies and policies for proactive data leakage prevention and incident response.

Module Topics:

1. Introduction to Data Leakage in Cloud Computing

- Definition and implications of data leakage

- Challenges and risks associated with data leakage in cloud environments

2. Types of Data Leakage Threats in Cloud Computing

- Insider threats

- External attacks

- Accidental leakage

- Malicious activity

3. Data Leakage Detection Techniques

- Signature-based detection

- Anomaly detection

- Machine learning-based approaches

- Behavior analysis

4. Data Leakage Prevention Mechanisms

- Encryption techniques

- Access controls and permissions management

- Role-based access control (RBAC)

- Multi-factor authentication (MFA)

5. Tools and Technologies for Data Leakage Detection

- Data loss prevention (DLP) solutions

- Intrusion detection and prevention systems (IDPS)

- Network monitoring tools

- Cloud security platforms

6. Case Studies and Practical Implementation

- Analysis of recent data leakage incidents in cloud environments

- Hands-on exercises using data leakage detection tools and platforms

7. Policy Development and Incident Response

- Formulating data leakage prevention policies

- Incident response procedures and best practices

- Continuous monitoring and improvement strategies

Assessment Methods:

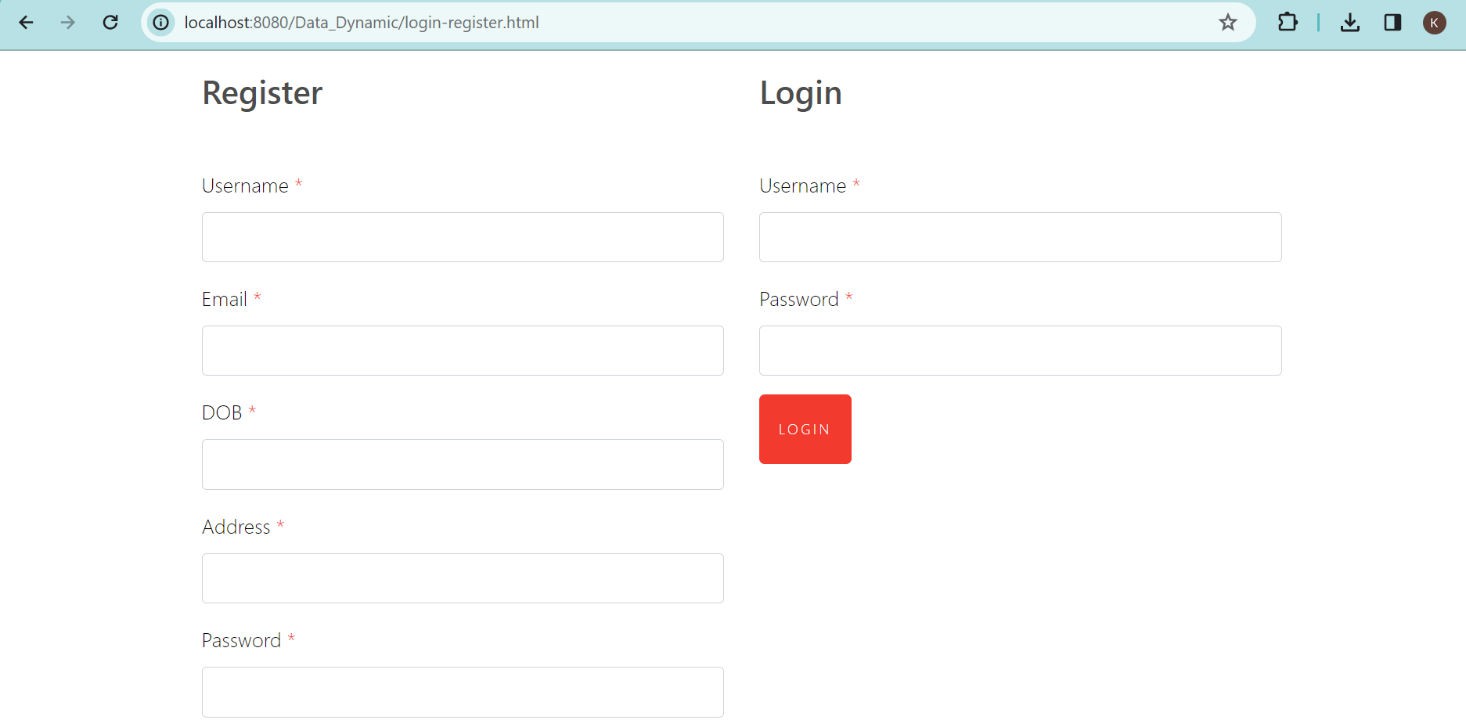
1. Written assignments or reports on data leakage detection techniques and case studies.

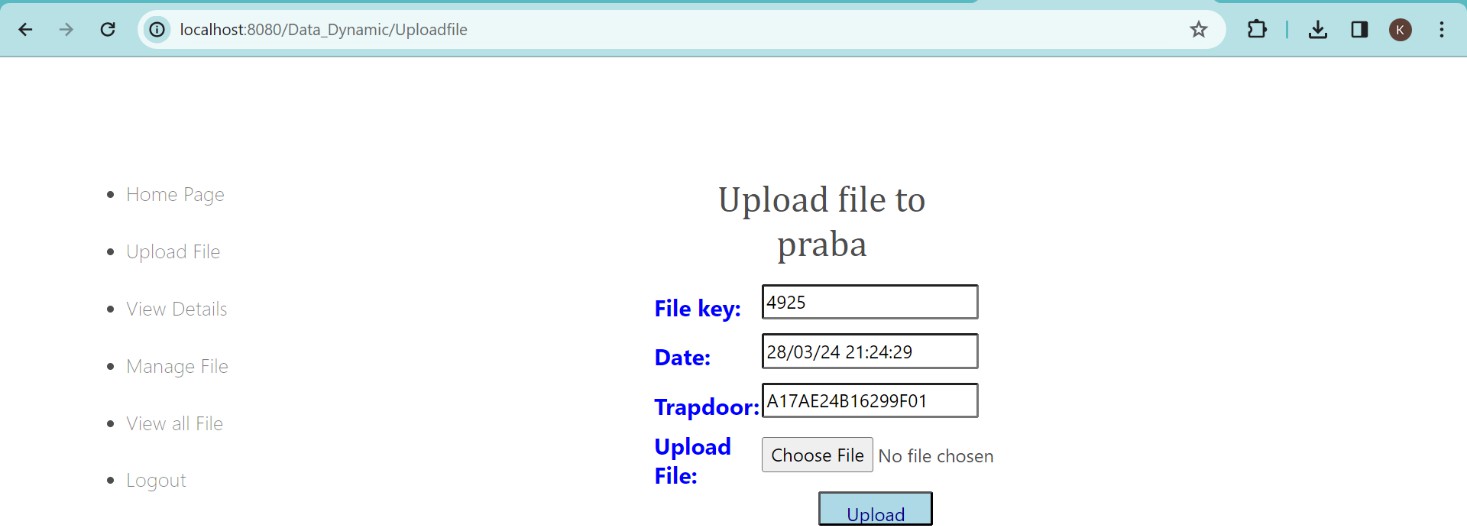
2. Practical exercises demonstrating the implementation of data leakage detection solutions.

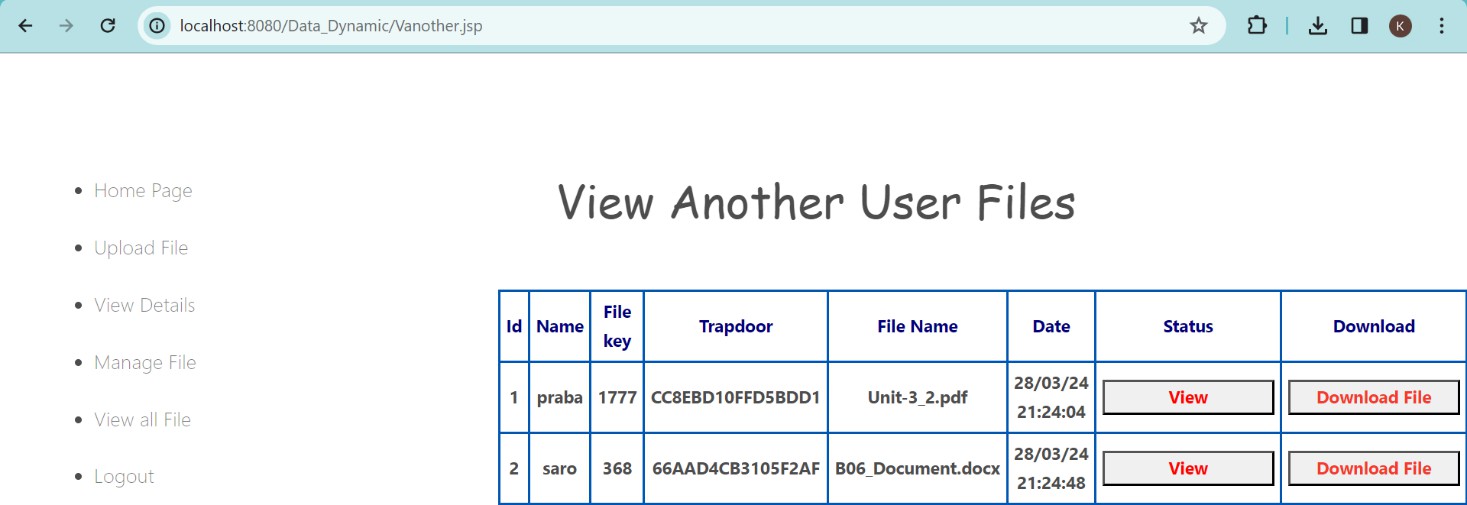
3. Group discussions and presentations on policy development and incident response strategies.

4. Examinations assessing theoretical knowledge and understanding of module topics.

**APPENDICES**

**7.1 SCREEN SHOTS**









**7.2 SOURCE CODE**

**/\***

**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'),('2','saro','4636','B04506D7857F2E06','READ ME.txt','Nodify send'),('4','ragu','2136','7DF59C8B0AD2DEB0','READ ME.txt','Nodify send'),('5','kingsleen','2449','1997A951B8F4D59C','file1.txt','Nodify send'),('5','kingsleen','2449','1997A951B8F4D59C','file1.txt','Nodify send'),('5','kingsleen','2449','1997A951B8F4D59C','file1.txt','Nodify send');**

**/\*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;**

**/\*Data for the table `request` \*/**

**insert into `request`(`id`,`name`,`fkey`,`trap`,`fname`,`fkkey`,`status`) values ('1','praba','6461','2ECB5F90560D91FA','READ ME.txt','6461','Send Key'),('2','saro','6461','2ECB5F90560D91FA','READ ME.txt','6461','Send Key'),('4','ragu','4636','B04506D7857F2E06','READ ME.txt','4636','Send Key'),('5','kingsleen','6461','2ECB5F90560D91FA','READ ME.txt','6461','Send Key'),('5','kingsleen','6461','2ECB5F90560D91FA','READ ME.txt','6461','Send Key');**

**/\*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,**

**`image` varchar(200) DEFAULT NULL,**

**`Attack` varchar(200) DEFAULT 'No'**

**) ENGINE=InnoDB DEFAULT CHARSET=latin1;**

**/\*Data for the table `upload` \*/**

**insert into `upload`(`reg`,`name`,`fkey`,`dates`,`trapdoor`,`image`,`Attack`) values ('1','praba','6461','14/09/20 18:54:25','2ECB5F90560D91FA','READ ME.txt','Attack'),('2','saro','4636','14/09/20 19:30:29','B04506D7857F2E06','READ ME.txt','Attack'),('4','ragu','2136','16/09/20 11:31:40','7DF59C8B0AD2DEB0','READ ME.txt','No'),('5','kingsleen','2449','21/09/20 16:35:10','1997A951B8F4D59C','file1.txt','Attack');**

**/\*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=6 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.com','9898998989','chhennai','Authorized'),(2,'saro','saro','21/01/1999','1cp.javateam2020@gmail.com','9389309389','chemmai','Authorized'),(3,'ram','ram','21/01/2007','ram@gmail.com','9829828922','chenni','Unauthorized'),(4,'ragu','ragu','21/01/2020','1cp.javateam2020@gmail.com','','kumbakonam','Authorized'),(5,'kingsleen','password','14/11/1982','kingsleen.singapore@gmail.com','9865133648','chennai','Authorized');**

**/\*!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 \*/;**

**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.

BIBLIOGRAPHY

**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 Al-Nabhani, et al. New comparative study between des, 3des and aes within nine factors. arXiv preprint arXiv:1003.4085, 2010.
10. Aman Kumar, Sudesh Jakhar, and Sunil Makkar. Distinction between secret key and public key cryptography with existing glitches. Indian Journal of Education and Information Management, 1(9):392–395, 2012.
11. Hitendra GARG and Suneeta AGARWAL. A secure image based watermarking for 3d polygon mesh. SCIENCE AND TECHNOLOGY, 16(4):287–303, 2013.
12. Hitendra Garg and Suneeta Agrawal. Uniform repeated insertion of redundant watermark in 3d object. In Signal Processing and Integrated Networks (SPIN), 2014 International Conference on, pages 184–189. IEEE, 2014.
13. CISSP Susan Hansche, CISSP John Berti, and Chris Hare. Official (ISC) 2 guide to the CISSP exam. CRC Press, 2003.
14. D Elliott Bell and Leonard J La Padula. Secure computer system: Unified exposition and multics interpretation. Technical report, DTIC Document, 1976.
15. David Elliott Bell. Looking back at the bell-la padula model. In ACSAC, volume 5, pages 337–351, 2005.
16. Fred´ eric Deguillaume, Sviatoslav V Voloshynovskiy, and Thierry Pun. ´ Method for the estimation and recovering from general affine transforms in digital watermarking applications. In Electronic Imaging 2002, pages 313–322. International Society for Optics and Photonics, 2002.
17. Stallings William and William Stallings. Cryptography and Network Security, 4/E. Pearson Education India, 2006.
18. Achal Kumar and Vibhav Prakash Singh. Digital watermarking using color image processing using images for transmitting secret information.
19. JJK RUANAIDH and T PUN. Rotation, scale and translation invariant spread spectrum digital image watermarking. Signal processing, 66(3):303–317, 1998.
20. NIST FIPS Pub. 197. Announcing the Advanced Encryption Standard (AES), 2001.