CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents. Then further data will be provided by the distributor to the trusted third party of the enterprise using this application. This application will further monitor if in case any data has been leaked by the agent of the enterprise.

1.2 BRIEF DESCRIPTION

This will give the description about the overall mathematical model of the project and set theory used in the project

1.2.1 Project Problem statement:

To build an application that helps in **Detecting the data** which has been leaked. It also helps in finding **Guilt of Agent** from the given set of agents which has leaked the data using **Probability Distribution**.

1.2.2 Problem description:

DLD is the system such that DLD={A,,D,T,U,R,S,U*,C,M,F}.

.

- {A} is the Administrator who will be given all the privileges about the application activities such as blocking the guilty agent using probability distribution table. He can also see the shared files and manages the whole database
- {D} is the Distributor who will send data T to different agents U .He will also accept requests from the different agents about the particular type of files and will distribute it accordingly
- T is the set of Data object that are supplied to agents. T can be of any type and size, e.g., they could be tuples in a relation, or relations in a database. Specifically in our project we are sending the text files.

  $T= \{t_1,t_2,t_3,...t_n\}$

- U is the set of Agents who will receive the data from the distributor D. The agent can also request for the particular type of text files. These agents are continuously monitored by the administrator and distributor in case they leak the data.

  $U=\{u_1,u_2,u_3,...u_n\}$

- R is the Record set of data objects which is sent to agents. This data is sent to agent depending upon the distributor’s choice or agent’s request.

  $R=\{t_1,t_3,t_5..tm\}$ **R is a subset of T**

- S is the Record set of data objects which are leaked by the particular agent U_i to third party. This agent who has leaked the data will be further referred as a guilty agent and will be blocked by the admin.

  $S=\{t_1,t_3,t_5..tm\}$ **S is a Subset of T**

- U* is the set of all agents which may have leaked the data. They are referred as Guilty Agents

  $U^*=\{u_1,u_3,...u_m\}$ **U* is a subset of U**

- C is the set of Conditions which will be given by the agents to the distributor

  $C=\{cond_1,cond_2,cond_3,...,cond_n\}$

- M is set of data objects to be send in Sample Data Request algorithm
  $M=\{m_1,m_2,m_3,...,m_n\}$

- F is the set of fake objects that are sent along with the original objects.
  $F=\{f_1,f_2,f_3...,f_n\}$

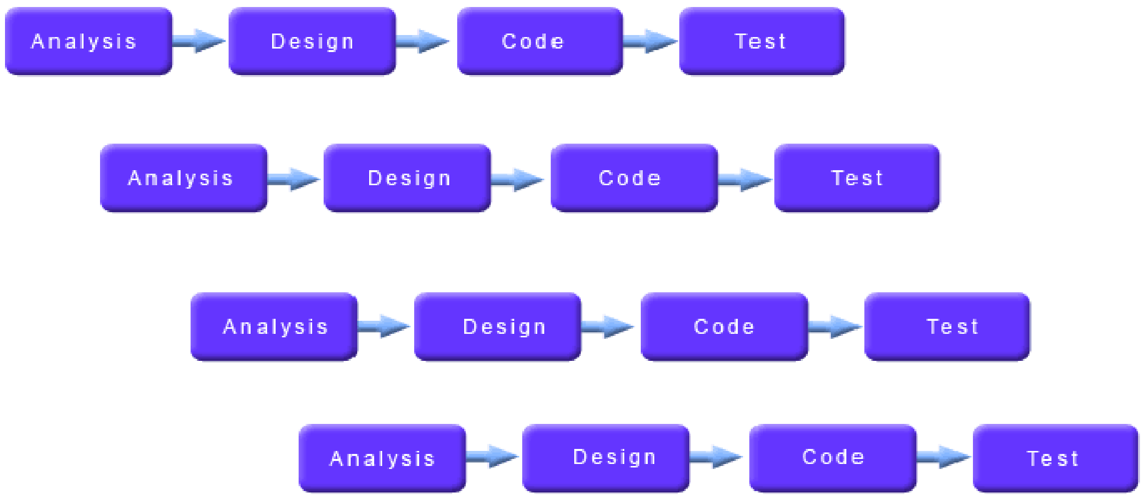
1.3 PROBLEM DEFINITION

Our goal is to detect when the distributor's sensitive data have been leaked by agents, and if possible to identify the agent that leaked the data.

1.4 APPLYING SOFTWARE ENGINEERING APPROACH

Incremental model is an evolution of waterfall model. The product is designed, implemented, integrated and tested as a series of incremental builds. It is a popular model software evolution used many commercial software companies and system vendor. Incremental software development model may be applicable to projects where:
Software Requirements are well defined, but realization may be delayed.

The basic software functionality are required early



Advantages

- Generates working software quickly and early during the software life cycle.
- More flexible - less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Easier to manage risk because risky pieces are identified and handled during its iteration.

Disadvantages

- Each phase of an iteration is rigid and do not overlap each other.
- Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.

CHAPTER 2

LITERATURE SURVEY

2.1 DATA LEAKAGE DETECTION

We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. For example, one can add random noise to certain attributes, or one can replace exact values by ranges. However, in some cases it is important not to alter the original distributor's data. For example, if an outsourcer is doing our payroll, he must have the exact salary and customer bank account numbers. If medical researchers will be treating patients (as opposed to simply computing statistics), they may need accurate data for the patients. Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. In this paper we study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a web site, or may be obtained through a legal discovery process.)

At this point the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with 5 cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings. In this paper we develop a model for assessing the “guilt” of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker.

2.2 DATA MINING

Data Mining (the analysis step of the **knowledge discovery in databases** process, or KDD), a relatively young and interdisciplinary field of computer science is the process of discovering new patterns from large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics and database systems. The goal of data mining is to extract knowledge from a data set in a human-understandable structure and involves database and data management, data preprocessing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of found structure, visualization and online updating.

2.2.1 The Scope of Data Mining

Data mining derives its name from the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing these capabilities:

- **Automated prediction of trends and behaviors**. Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data — quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

- **Automated discovery of previously unknown patterns**. Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

Data mining techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems as existing platforms are upgraded and new products developed. When data mining tools are implemented on high performance parallel processing systems, they can analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyze huge quantities of data. Larger databases, in turn, yield improved predictions.

2.3 WATERMARKING

Digital Watermarking describes methods and technologies that hide information, for example a number or text, in digital media, such as images, video or audio. The embedding takes place by manipulating the content of the digital data, which means the information is not embedded in the frame around the data. The hiding process has to be such that the modifications of the media are imperceptible. For images this means that the modifications of the pixel values have to be invisible. Furthermore, the watermark must be either robust or fragile, depending on the application. By "robust" we mean the capability of the watermark to resist manipulations of the media, such as lossy compression (where compressing data and then decompressing it retrieves data that may well be different from the original, but is close enough to be useful in some way), scaling, and cropping, just to enumerate some. In some cases the watermark may need to be fragile. "Fragile" means that the watermark should not resist tampering, or would resist only up to a certain, predetermined extent

The first applications that came to mind were related to copyright protection of digital media. In the past duplicating art work was quite complicated and required a high level of expertise for the counterfeit to look like the original. However, in the digital world this is not true. Now it is possible for almost anyone to duplicate or manipulate digital data and not lose data quality. Similar to the process when artists creatively signed their paintings with a brush to claim copyrights, artists of today can watermark their work by hiding their name within the image. Hence, the embedded watermark permits identification of the owner of the work. It is clear that this concept is also applicable to other media such as digital video and audio. Currently the unauthorized distribution of digital audio over the

Internet in the MP3 format is a big problem. In this scenario digital watermarking may be useful to set up controlled audio distribution and to provide efficient means for copyright protection, usually in collaboration with international registration bodies.

2.4 KNOWLEDGE BASED SYSTEM

Knowledge based systems are artificial intelligent tools working in a narrow domain to provide intelligent decisions with justification. Knowledge is acquired and represented using various knowledge representation techniques rules, frames and scripts. The basic advantages offered by such system are documentation of knowledge, intelligent decision support, self learning, reasoning and explanation. **Knowledge-based systems** are systems based on the methods and techniques of Artificial Intelligence. Their core components are:

- knowledge base
- acquisition mechanisms
- inference mechanisms

Knowledge Base Systems (KBS) goes beyond the decision support philosophy to indicate the expert system technology into the decision making framework. Expert Systems (ES) have been the tools and techniques perfected by artificial intelligence (AI) researchers to deduce decision influences based on codification of knowledge. The codification of knowledge uses the principles of knowledge representation (part of the large theoretical ideas of knowledge engineering). Typically such codification uses rules like IF-THEN rules to represent logical implications.

2.5 ACHIEVING K-ANONYMITY PRIVACY PROTECTION USING GENERALIZATION

Today’s globally networked society places great demand on the collection and sharing of person-specific data for many new uses. This happens at a time when more and more historically public information is also electronically available. When these data are linked together, they provide an electronic image of a person that is as identifying and personal as a fingerprint even when the information contains no explicit identifiers, such as name and phone number. Other distinctive data, such as birth date and postal code, often combine uniquely and can be linked to publicly available information to re-identify individuals.

So in today's technically-empowered data rich environment, how does a data holder, such as a medical institution, public health agency, or financial organization, share person-specific records in such a way that the released information remain practically useful but the identity of the individuals who are the subjects of the data cannot be determined? One way to achieve this is to have the released information adhere to k-anonymity. A release of data is said to adhere to k-anonymity if each released record has at least (k-1) other records also visible in the release whose values are indistinct over a special set of fields called the quasi-identifier. The quasi-identifier contains those fields that are likely to appear in other known data sets. Therefore, k-anonymity provides privacy protection by guaranteeing that each record relates to at least k individuals even if the released records are directly linked (or matched) to external information. This paper provides a formal presentation of achieving k-anonymity using generalization and suppression. Generalization involves replacing (or recoding) a value with a less specific but semantically consistent value. Suppression involves not releasing a value at all. While there are numerous techniques available2, combining these two offers several advantages. First, a recipient of the data can be told what was done to the data. This allows results drawn from released data to be properly interpreted. Second, information reported on each person is "truthful" which makes resulting data useful for fraud detection, counter-terrorism surveillance, healthcare outcome assessments and other uses involving traceable person-specific patterns3. Third, these techniques can provide results with guarantees of anonymity that are minimally distorted. Any attempt to provide anonymity protection, no matter how minor, involves modifying the data and thereby distorting its contents, so the goal is to distort minimally. Fourth, these techniques can be used with preferences a recipient of the released data may have, thereby providing the most useful data possible. In this way, algorithmic decisions about how to distort the data can have minimal impact on the data's fitness for a particular task.

2.6 ENCRYPTION ALGORITHM (AES)

The Advanced Encryption Standard (AES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in December 2001.

In general the block size and the key length can be any of the five allowed values but for the Advanced Encryption Standard (AES) the block size is fixed at 128 bits and the key length can only be 128, 192 or 256 bits.

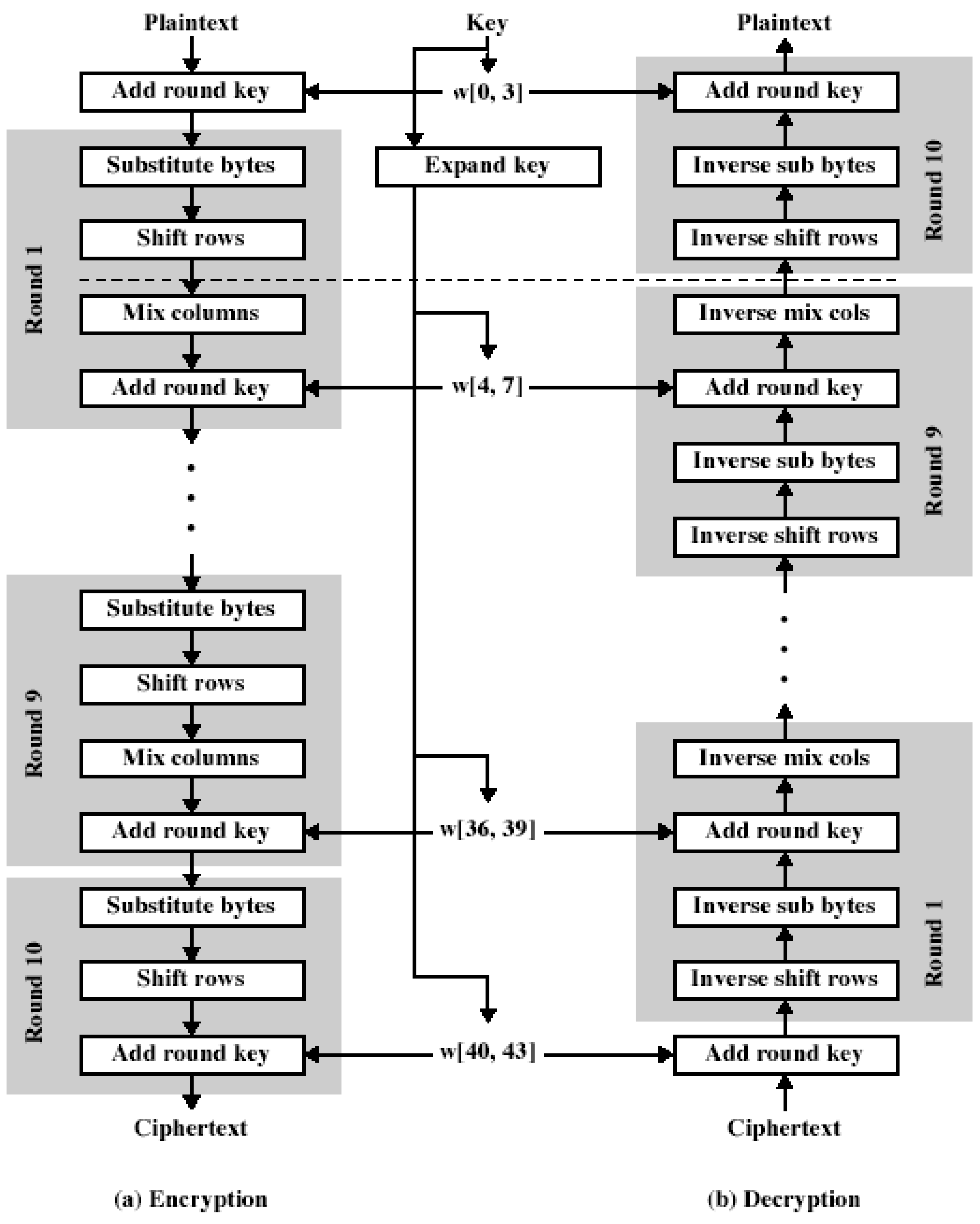


FIG: 2.1 AES Algorithm

2.6.1 Comparison between AES and DES:

DES (Data Encryption Standard) is a rather old way of encrypting data so that the information could not be read by other people who might be intercepting traffic. DES is rather quite old and has since been replaced by a newer and better AES (Advanced Encryption Standard). The replacement was done due to the inherent weaknesses in DES that allowed the encryption to be broken using certain methods of attack. Common applications of AES, as of the moment, are still impervious to any type of cracking techniques, which makes it a good choice even for top secret information.

The inherent weakness in DES is caused by a couple of things that are already addressed in AES. The first is the very short 56 bit encryption key. The key is like a password that is necessary in order to decrypt the information. A 56 bit has a maximum of 256 combinations, which might seem like a lot but is rather easy for a computer to do a brute force attack on. AES can use a 128, 192, or 256 bit encryption key with 2^128, 2^192, 2^256 combinations respectively. The longer encryption keys make it much harder to break given that the system has no other weaknesses.

Another problem is the small block size used by DES, which is set at 64 bits. In comparison, AES uses a block size that is twice as long at 128 bits. In simple terms, the block size determines how much information you can send before you start having identical blocks, which leak information. People can intercept these blocks and use read the leaked information. For DES with 64 bits, the maximum amount of data that can be transferred with a single encryption key is 32GB; at this point another key needs to be used. With AES, it is at 256 exabytes or 256 billion gigabytes. It is probably safe to say that you can use a single AES encryption key for any application.

In terms of structure, DES uses the Feistel network which divides the block into two halves before going through the encryption steps. AES on the other hand, uses permutation-substitution, which involves a series of substitution and permutation steps to create the encrypted block.

2.7 PRIVACY, PRESERVATION AND PERFORMANCE (THE 3 P'S)

There are several attributes to consider when designing a dis-tributed data management system. For example, we would like a system that enforces "privacy" by not divulging data to unau-thorized entities. At the same time, we want to "preserve" the data effectively i.e., protect it from hardware failures, natural disasters, and so on. And, of course, we do not want to sacri-fice "performance" – a system that runs slowly may not be very useful. There are many such desirable attributes: confidential-ity, integrity, reliability, availability, throughput, and so on.

While each of these attributes has been studied extensively, there is not much work that considers the tradeoffs between them. We believe that today the real challenge in designing a system is in achieving a balance between several conflicting at-tributes. For example, a system that simply deletes all input data would be very “secure” – no data will ever leak out! – but would be unattractive in other dimensions. Similarly, one can build a highly “reliable” system that proliferates many copies of its data. This system would excel along the preservation dimension, but each extra copy would increase the chances of unau-thorized break-ins. If we encrypt a large collection of records as a single “blob”, performance for reading individual records suffers. If we encrypt each record individually, reads will be faster, but overall security may not be as strong.

In this paper we study, in a unified way, three conflicting attributes of a distributed system, and show how to design systems that strike a balance among them. We refer to these attributes as the “3 P’s” of distributed data management – privacy, preservation and performance. Informally:

- Privacy refers to protecting data from unauthorized access (related to security and confidentiality).
- Preservation ensures that data is still available and uncorrupted far into the future (related to integrity, availability, and reliability).
- Performance refers to the quick, timely access to our distributed data (related to response time and throughput).

The main contributions are:

A unified framework for measuring the privacy, preservation and performance offered by a system.

1. An algorithm for finding systems that achieve a desired balance between the 3 P’s.

Our work represents a bridge between system design and risk management. Risk management is the science of identifying, measuring and mitigating the uncertainty related to threats.

We identify privacy violations, data loss and poor performance as threats faced by a system in its daily operation. We then measure the potential harm caused by these threats:

a) Privacy is measured by the damage, D, caused by leakage of in-formation to unauthorized parties,

b) Preservation is measured by the loss, L, incurred due to lost or corrupted data, and

c) Performance is measured by the running time, T, of read and write commands issued to the system.

We mitigate risk not by providing “guarantees” – instead, we accept that bad things can happen, and try to minimize the harm when bad things do happen. How can we ensure that we do well “on average”, and how can we reduce the likelihood of “catastrophes”? We treat D, L and T as random variables, and solve optimization problems involving their means and variances.

To effectively manage risk, it is crucial to account for the heterogeneity within a collection of data objects. For exam-ple, there may be a great deal of damage done if an internal e-mail discussing corporate strategy is leaked to outsiders. But, nobody may care if the e-mail was accidentally deleted. On the other hand, nobody would mind if marketing materials were “leaked” to outsiders (it’s probably a good thing!). But if these marketing materials were lost, the time and money spent developing them would be lost. Moreover, e-mail is mostly text whereas marketing flyers are filled with large images, so the performance implications are vastly different between the two. In our framework, we explicitly model data objects in a heterogeneous collection as having differing values, sensitivities, sizes and usage patterns.

The outcome of our work is a solution strategy for the following problem: Given a set of resources (e.g., data centers, servers, storage devices), and a collection of data objects (e.g., documents, photos, MP3s, e-mail) with known usage characteristics, sensitivities and sizes, how should we distribute them across our resources to achieve a desired balance between the 3 P’s – privacy, preservation and performance?

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 INTRODUCTION

In this project, we develop a model for assessing the “guilt” of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. Finally, we also consider the option of adding “fake” objects to the distributed set. Such objects do not correspond to real entities but appear realistic to the agents. In a sense, the fake objects act as a type of watermark for the entire set, without modifying any individual members. If it turns out that an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.

We define unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study the following scenario: After giving a set of objects to agents, thedistributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a website, or may be obtained through a legal discovery process.) At this point, the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means.

Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with five cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees “enough evidence” that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.

3.1.1 Objective

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents.

Our goal is to detect when the distributor's sensitive data have been leaked by agents, and if possible to identify the agent that leaked the data.

3.1.2 Project Scope

Aim of the project is to introduce an easy and efficient solution which is capable of detecting data leakage in the system and if possible the agent details who did it:

- Agent/Distributor Data Communication
- Finding Guilty Agent and their details

We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data are modified and made less sensitive before being handed to agents. For example, one can add random noise to certain attributes, or one can replace exact values by ranges. However, in some cases, it is important not to alter the original distributor's data. For example, if an outsourcer is doing our payroll, he must have the exact salary and customer bank account numbers. If medical researchers will be treating patients (as opposed to simply computing statistics), they may need accurate data for the patients.

we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a website, or may be obtained through a legal discovery process.) At this point, the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with five cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.

3.1.3 User Classes and Characteristics

- Administrator who will be given all the privileges about the application activities such as blocking the guilty agent using probability distribution table. He can also see the shared files and manages the whole database
- Distributor who will send data T to different agents U .He will also accept requests from the different agents about the particular type of files and will distribute it accordingly
- Agents who will receive the data from the distributor D. The agent can also request for the particular type of text files. These agents are continuously monitored by the administrator and distributor in case they leak the data.

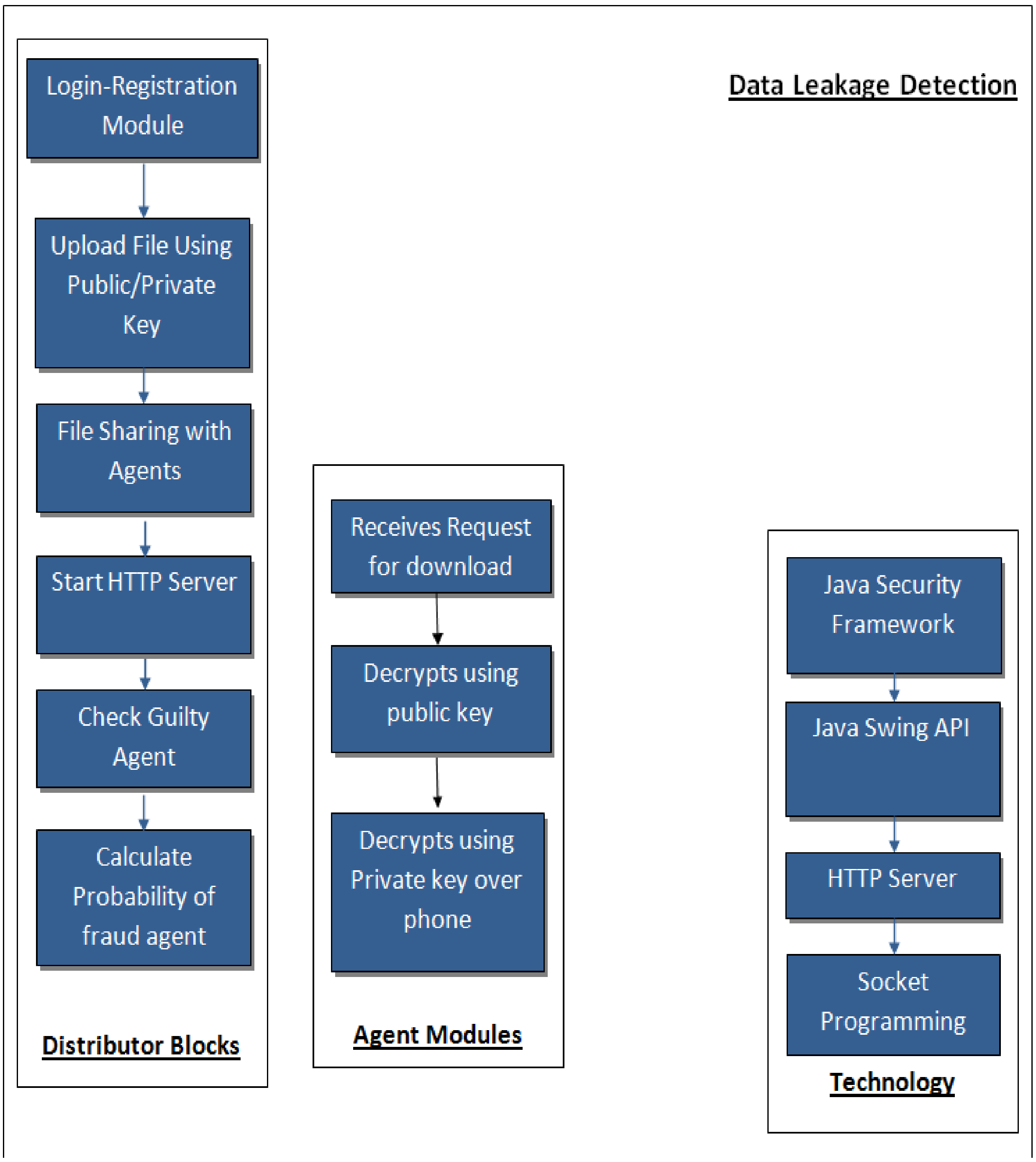

Data Leakage Detection
Login-Registration Module
Upload File Using Public/Private Key
File Sharing with Agents
Start HTTP Server
Check Guilty Agent
Calculate Probability of fraud agent
Distributor Blocks
Receives Request for download
Decrypts using public key
Decrypts using Private key over phone
Agent Modules
Java Security Framework
Java Swing API
HTTP Server
Socket Programming
Technology

FIG: 3.1.2 Data Leakage Detection Overview

3.1.4 Operating Environment

To develop this system, we are using JAVA language and its different technology.

3.1.4.1 Java Swing API

Swing is the primary Java GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check box and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables and lists.

Unlike AWT components, Swing components are not implemented by platform-specific code. Instead they are written entirely in Java and therefore are platform-independent. The term "lightweight" is used to describe such an element.

3.1.4.2 HTTP Server

A **Http Server** is bound to an IP address and port number and listens for incoming TCP connections from clients on this address. The sub-class Https Server implements a server which handles HTTPS requests.

The HTTP Server API enables applications to communicate over HTTP without using Microsoft Internet Information Server (IIS). Applications can register to receive HTTP requests for particular URLs, receive HTTP requests, and send HTTP responses. The HTTP Server API includes SSL support so that applications can exchange data over secure HTTP connections without IIS. It is also designed to work with I/O completion ports.

3.1.4.3 Socket Programming

A socket is an endpoint of a two-way communication link between two programs running on the network.

Socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent. Java provides a set of classes, defined in a package called java.net, to enable the rapid development of network applications. Key classes, interfaces, and exceptions in java.net package simplifying the complexity involved in creating client and server programs

3.1.5 Product Features

Product features are:

1. Agent/Distributor Sign up/Login
2. Distributor distributes files to agents
3. File Encryption using public and private key
4. Guilty Agent Report and their machine details
5. Agent requests for the particular text file.
6. Probability distribution of the agents.
7. Administrator can block the guilty agent.

3.1.6 Design and Implementation Constraints

Design Constraints

a. Error Recognition: Error should be easily recognized and get solved out.

General Constraints

a. Network Speed – Files needs to transmit for any transaction using triple AES encryption, it introduces an additional overhead on network bandwidth

User Documentation

a. Currently we will be using Java to develop Data Leakage detection techniques.

b. Files will be public/private key encrypted.

3.1.7 Assumptions and Dependencies

Assumptions:

a. Agent must have basic knowledge of internet.

b. Agent should login from single workstation/machine.

Dependencies:

a. Speed of the network

b. Size of the file

3.2. SYSTEM FEATURES

The main focus of this project is the data allocation problem: how can the distributor "intelligently" give data to agents in order to improve the chances of detecting a guilty agent?

As illustrated in Fig. 2, there are four instances of this problem we address, depending on the type of data requests made by agents and whether "fake objects" are allowed.

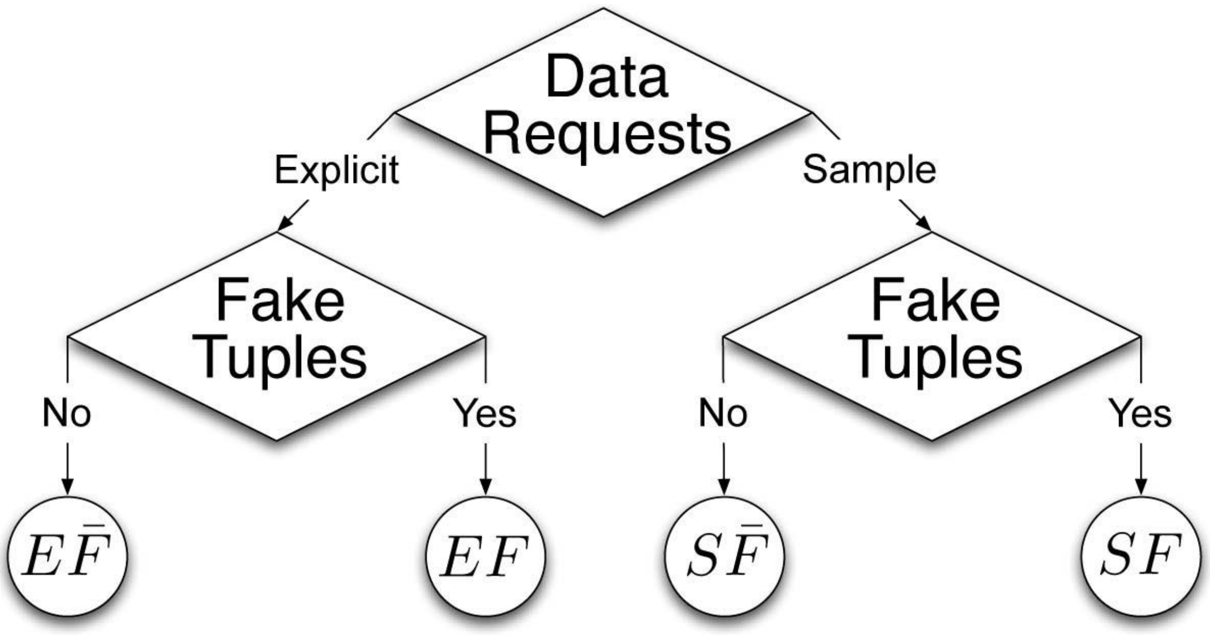


FIG: 3.2 System Feature

The two types of requests we handle were defined in Section 2: sample and explicit. Fake objects are objects generated by the distributor that are not in set T. The objects are designed to look like real objects, and are distributed to agents together with T objects, in order to increase the chances of detecting agents that leak data.

The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. However, fake objects may impact the correctness of what agents do, so they may not always be allowable. The idea of perturbing data to detect leakage is not new, e.g., However, in most cases, individual objects are perturbed, e.g., by adding random noise to sensitive salaries, or adding a watermark to an image. In our case, we are perturbing the set of distributor objects by adding fake elements. In some applications, fake objects may cause fewer problems that perturbing real objects.

For example, say that the distributed data objects are medical records and the agents are hospitals. In this case, even small modifications to the records of actual patients may be undesirable. However, the addition of some fake medical records may be acceptable, since no patient matches these records, and hence, no one will ever be treated based on fake records. Our use of fake objects is inspired by the use of “trace” records in mailing lists.

In this case, company A sells to company B a mailing list to be used once (e.g., to send advertisements). Company A adds trace records that contain addresses owned by company A. Thus, each time company B uses the purchased mailing list, A receives copies of the mailing. These records are a type of fake objects that help identify improper use of data. The distributor creates and adds fake objects to the data that he distributes to agents. We let Fi -Ri be the subset of fake objects that agent Ui receives. As discussed below, fake objects must be created carefully so that agents cannot distinguish them from real objects. In many cases, the distributor may be limited in how many fake objects he can create. For example, objects may contain e-mail addresses, and each fake e-mail address may require the creation of an actual inbox (otherwise, the agent may discover that the object is fake). The inboxes can actually be monitored by the distributor: if e-mail is received from someone other than the agent who was given the address, it is evident that the address was leaked. Since creating and monitoring e-mail accounts consumes resources, the distributor may have a limit of fake objects. If there is a limit, we denote it by B fake objects.

Similarly, the distributor may want to limit the number of fake objects received by each agent so as to not arouse suspicions and to not adversely impact the agents' activities. Thus, we say that the distributor can send up to bi fake objects to agent Ui.

The distributor's data allocation to agents has one constraint and one objective. The distributor's constraint is to satisfy agents' requests, by providing them with the number of objects they request or with all available objects that satisfy their conditions. His objective is to be able to detect an agent who leaks any portion of his data. We consider the constraint as strict. The distributor may not deny serving an agent request and may not provide agents with different perturbed versions of the same objects. We consider fake object distribution as the only possible constraint relaxation. Our detection objective is ideal and intractable. Detection would be assured only if the distributor gave no data object to any agent (discuss that to attain "perfect" privacy and security, we have to sacrifice utility). We use instead the following objective: maximize the chances of detecting a guilty agent that leaks all his data objects.

3.2.1 Data Allocation Problem

The main focus of this project is the data allocation problem: how can the distributor "intelligently" give data to agents in order to improve the chances of detecting a guilty agent?

1. Distributor will be sending data that will be encrypted using public and private key to agents; any agent who wants to download the file has to login in the system and download the file.
2. As the File is in encrypted format, agent will have to enter the private key to open the file in software.
3. Downloaded file cannot be opened using file system; it can only be viewed from the software only.

3.2.2 Optimization Module

The distributor's data allocation to agents has one constraint and one objective. The distributor's constraint is to satisfy agents' requests, by providing them with the number of objects they request or with all available objects that satisfy their conditions. His objective is to be able to detect an agent who leaks any portion of his data or lock the file so that it cannot be distributed or leaked.

3.2.3 Probability Calculation For Guilty Agent

Once we know that the file was leaked, we will have to calculate the probability of the agent being guilty.

3.3 EXTERNAL INTERFACE REQUIREMENTS

To identify and document the interfaces to other systems and external entities within the project scope.This is used to estimate the effort required to identify and document the interfaces to other systems and external entities within the project scope. The effort to identify and document the external interface requirements includes defining business elements for each external interface

3.3.1 User Interfaces

System will have following interfaces

- **Java Swing Client:** User upload/download module with public/private key encryption, login/registration module.
- **Java HTTP Server-** It will be used for sending/receiving files from distributor to agent and agent to supposedly trusted parties
- **Java Encryption/Decryption module-** Java based encryption/decryption using triple DES technique is used

3.3.2 Hardware Interfaces

- 2.4 GHZ, 80 GB HDD for installation.
- 512 MB memory.
- Users can use any PC based browser clients with IE 5.5 upwards. Internet service provider for group mailing with static IP.

3.3.3 Software Interfaces

- JDK 1.6
- Java Swing
- Netbeans 6.5
- Socket programming
- Triple DES algorithm

3.4 NONFUNCTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that define specific behavior or functions. The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.

3.4.1 Performance Requirements

System should process all internet banking requests in parallel for various users to give quick response time and should complete the process as a whole at one go.

UI should not hang for any enquiry from client.

3.4.2 Safety Requirements

The data safety must be ensured by arranging for a secure and reliable transmission media. The source and destination information must be entered correctly to avoid any misuse or malfunctioning.

3.4.3 Security Requirements

1. User password must be stored in encrypted form for the security reason
2. All the user details shall be accessible to only high authority persons.
3. Access will be controlled with usernames and passwords.
4. Files cannot be accessed from file system, only way to view is through private key decryption from software.

3.4.4 Software Quality Attributes

Reliability:

Reliability should be gained by making efficient database updates and taking backup regularly.

Availability :

The Data transfer is possible continuously between Distributor and Agent.

Flexibility :

Flexible service based architecture will be highly desirable for future extension.

3.5 OTHER REQUIREMENT

3.5.1 Database Requirements

- Database mission critical: no data loss and little (if any) downtime can be tolerated
- Need Backup and Recovery: no data backup is needed
- Data Types: relational data
- Need Replication: Database run from a single server so no need of Replication
- Platform Dependency: Database is platform independence
- Concurrent users will have database access: any number of users

3.5.2 Internalization Requirement

- Internationalization:- The provision within a computer program of the capability of making itself adaptable to the requirements of different native languages, local customs and coded character sets.
- Locale: - The definition of the subset of a user's environment that depends on language and cultural conventions.
- Localization: - The process of establishing information within a computer system specific to the operation of particular native languages, local customs and coded character sets.
- Globalization: - A product development approach which ensures that software products are usable in the worldwide markets through a combination of internationalization and localization.

3.5.3 Legal Requirements

Legal requirements for any software depend on the industry and type of business that particular software is engaged in. There is no one rule that fits all software. This will cover general legal requirements applicable to most software. You should consult a specialist to get a better understanding of your software legal requirements

Piracy Control

Your software cannot be designed to promote piracy of other software and/or websites on the Internet. You may face legal actions by various law enforcement agencies, for-profit companies and not-for-profit organizations if your software is found to promote piracy.

Patent Infringement and Copyright laws

You software cannot infringe on existing patents and violate copyright laws without written consent of all involved parties. Patent and copyright laws are different in each country, and if your software will be sold/used globally, you will have to adhere to all local patent and copyright laws.

Privacy Laws

If your software at any time records private information about your users, you have to let the users know what you plan do with that information. Some software companies do not share that information with third parties and they let users know about this to build more trust. Basically, your software cannot sell or use your users' information without their permission.

Safety Instructions

- You must let the users know of any known issues about your software that may be harmful for the users' computers or for users themselves. Failure to do so may result in lawsuits and complaints from users and other consumer protection agencies. You should also detail instructions to safely use your software.

3.6 ANALYSIS MODELS

3.6.1 Data Flow Diagrams

Level 0:

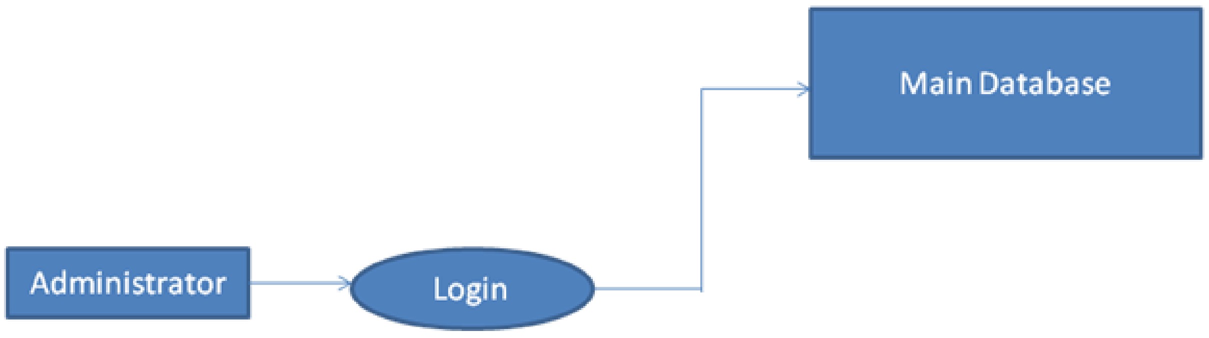


Fig 3.6.1A: Data Allocation

Level 1:

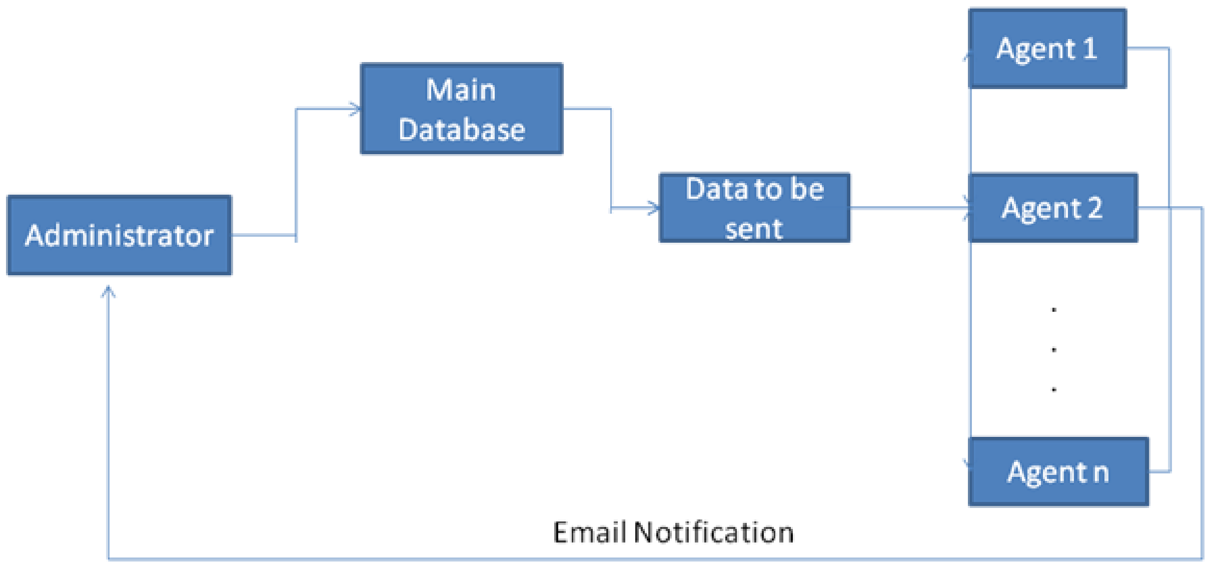


Fig 3.6.1B: Data Distribution

Level 2:

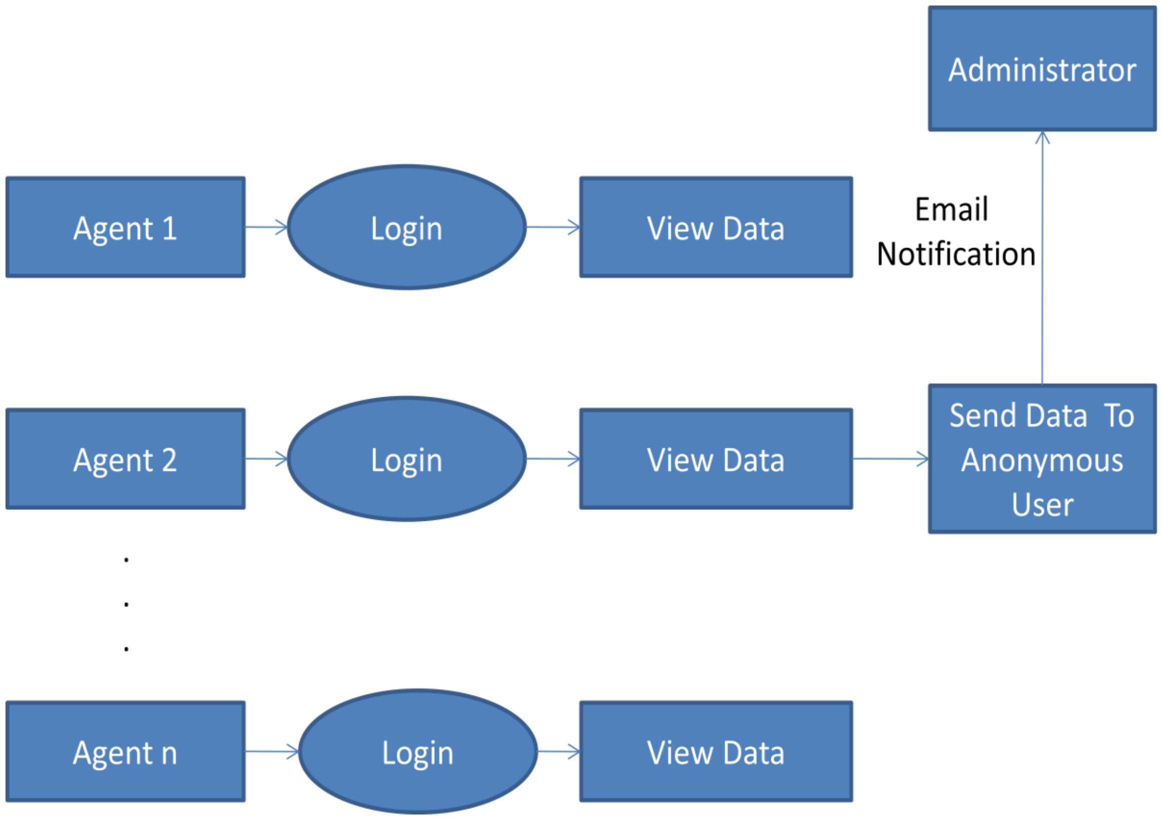


Fig 3.6.1C: Data Leakage and Detection

3.6.2 Class Diagrams

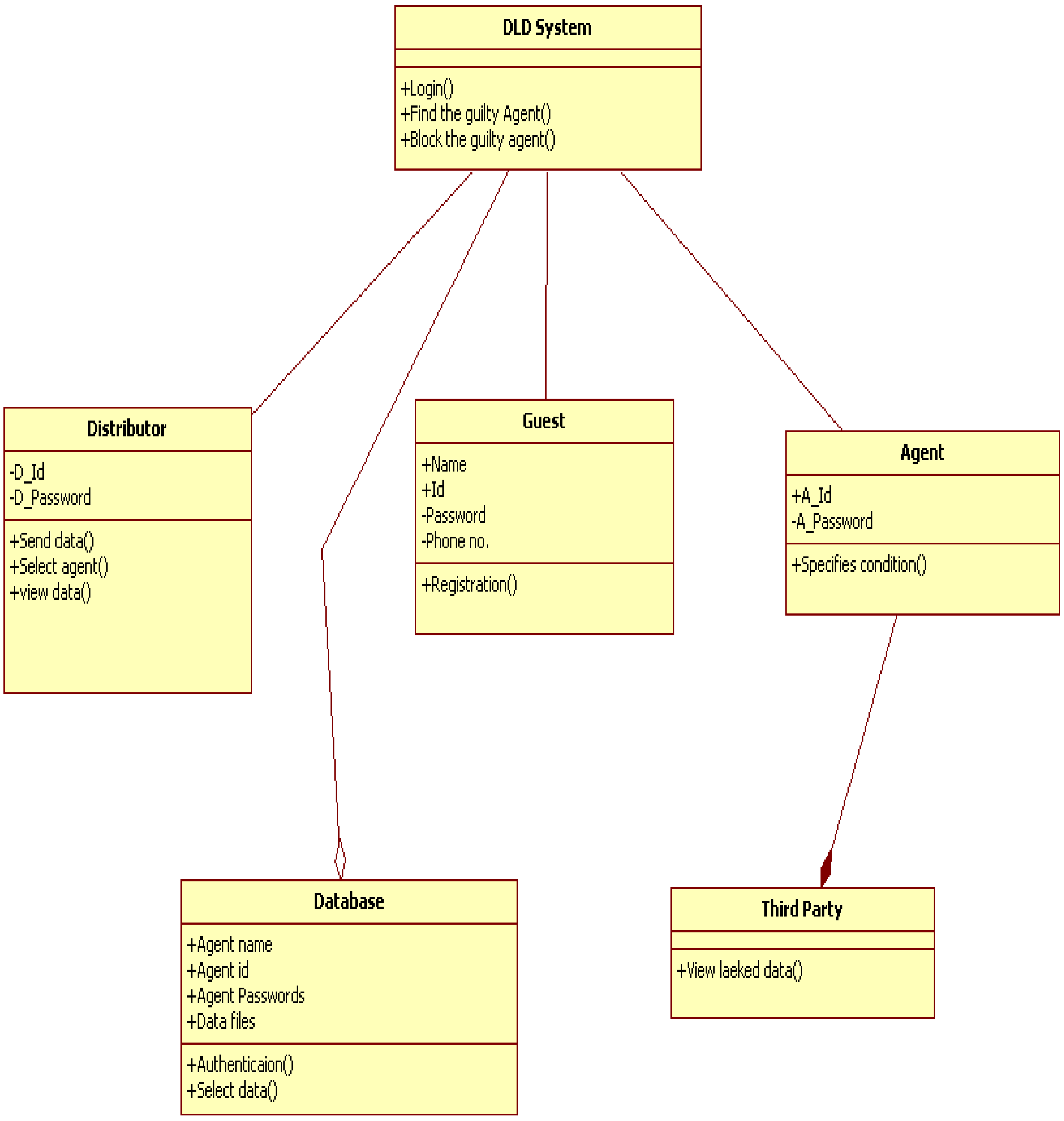


FIG3.6.2: Class Diagram

3.6.3 ENTITY RELATIONSHIP DIAGRAM

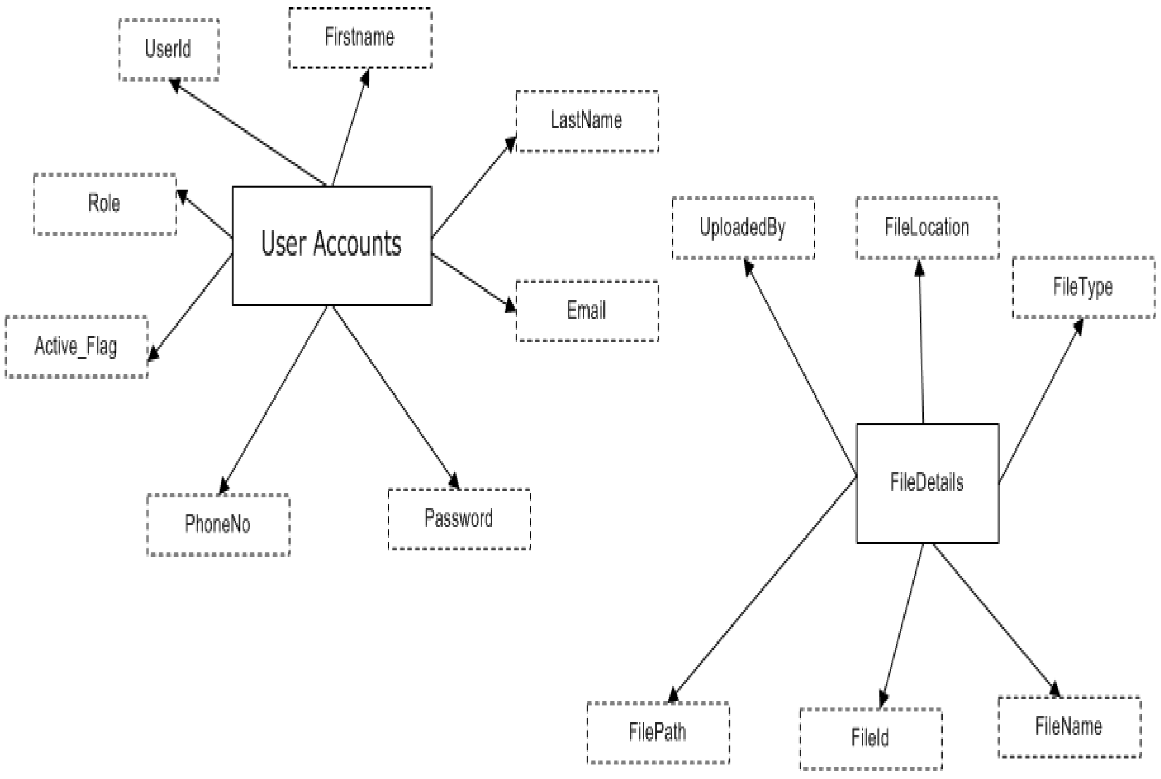


FIG3.6.3:E-R Diagram

ER Diagram shows the relationship between the entities and attributes. In this diagram there are two entities named user account and file details. The attributes for User accounts are Role, First name, Last name, user id etc and the attributes related to File Details Entity are File type, file path, file name.

3.7 SYSTEM IMPLEMENTATION PLAN

Month	**Schedule**	**Project task**
July	9-7-2011	Search for Topics on IEEE xplore.
	11-7-2011	Data Leakage Detection Topic Finalized
	12-7-2011	Papers on Data Leakage Detection searched.
	15-7-2011	Data collected on Data Leakage Detection
	20-7-2011	Preparation of Abstract & Synopsis.
	25-7-2011	Submission of Synopsis.
	29-7-2011	Guide Allocation & Literature Survey.
August	2-8-2011	List of Project Groups with Guide Names Displayed.
	5-8-2011	Meeting with guide about Mathematical Model about Project Statement.
	8-8-2011	Start of Preparation for Presentation.
	10-8-2011	First presentation with Guide about idea of project.
	16-8-2011	Requirement Analysis (SRS Document) Preparation.
	24-8-2011	Meeting with guide about preparation of SRS
	29-8-2011	Submission of SRS.

September	6-9-2011	Jotted down all the functions required for implementation of project.
	10-9-2011	Syntactical structure of functions and data structures required for implementation of project are finalized.
	12-9-2011	Preparation of UML diagrams.
	19-9-2011	Meeting with guide for showing the finalized syntactical structure of functions.
	20-9-2011	Start of Preparation for Presentation-II
October	22-9-2011	Meeting with guide for showing presentation of project design(with diagrams).
	26-9-2011	Presentation –II with design
	28-9-2011	Documentation of SRS with UML diagrams.
	30-9-2011	Start for preparation of preliminary report.
	4-10-2011	Meeting with guide about preliminary report.
	10-10-2011	Meeting with guide about preliminary report
	17-10-2011	Completion of preliminary report.

November	8-11-2011	Submission-Prelim Report of the Project

December	2nd Week	University Exam on Preliminary Report

January	1st Week	Coding
	2nd Week	At lest 2 module should finish (30%) Total Work
February	1st Week	First demonstration on project work expected (60%) of total work
March	1st Week	Test Plan , Design & Installation
	3rd Week	Final Project Demonstration
	4th Week	Preparation of project report , Preparation of Installable Project & Manual
April	1st Week	Submission of Report (Final Submission)
May	2nd Week	Final University Examination

Table 3.7: System Implementation Plan

CHAPTER 4

SYSTEM DESIGN

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner the interaction between parts in minimal clearly specified. Design will explain software components in detail; this will help the implementation of the system.

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

EXPECTED INPUT

Input design is the process of converting user-originated inputs to computer-based formats. Input design is one of the most expensive phases of the operation of computerized system and is often the major problem of a system.

EXPECTED OUTPUT

Output design generally refers to the results and information that are generated by the system for many end-user; output is the main reason for developing the system based on which they evaluate the usefulness of the application.

4.1 System Architecture

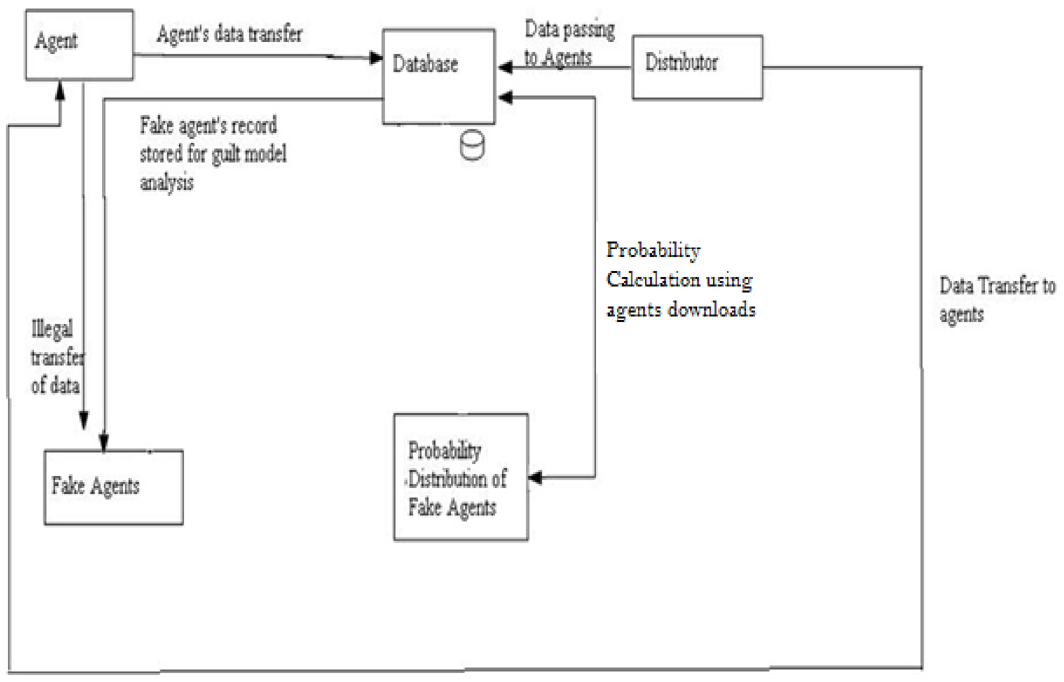

Agent
Agent's data transfer
Database
Data passing to Agents
Distributor
Fake agent's record stored for guilt model analysis
Probability Calculation using agents downloads
Data Transfer to agents
Illegal transfer of data
Fake Agents
Probability Distribution of Fake Agents

FIG 4.1: System Architecture

The working of the whole system is as follows:

1. The distributor logins into the system.

2. The distributor uploads the Data [eg.text files] into the Database.

4. Agent asks for the particular file or distributor uploads all the files for the agents accordingly

along with the private key after Login into the system.

5. Agents will download the files according to his needs [Sample request or Explicit request].

6. If any agents leak the data to the third party [Fake Agents] the distributor will check for the leaked data and will find the file which has been leaked.

7. Using probability distribution chart the Distributor will find the Agent which has leaked the data.

4.2 UML Diagrams

4.2.1 Use Case Diagram

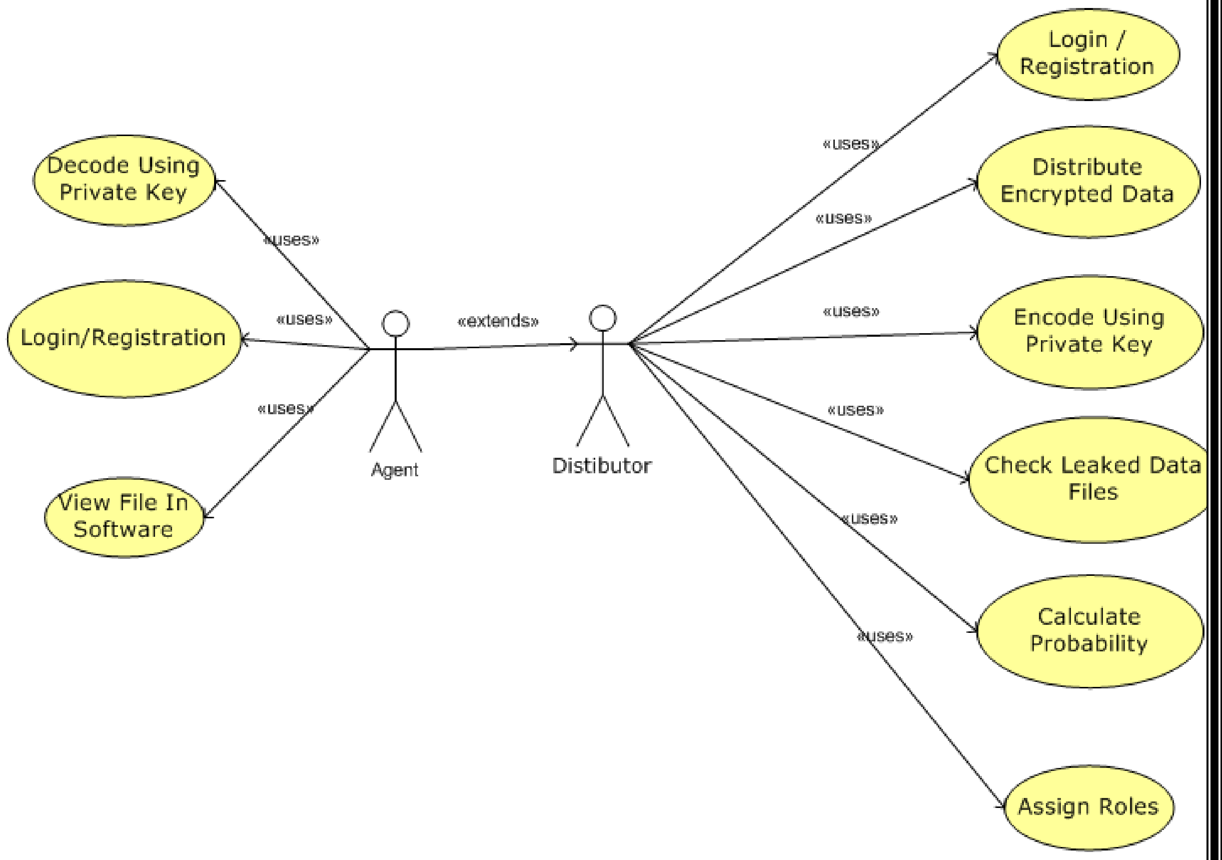


FIG 4.2.1: Use Case Diagram

Use Case Diagram:

What is use case diagram?

Use case diagram represents functionality of overall system.

There are mainly two parts-

1) Actor – who are going to perform the tasks.
2) Use case- which contains functions performed by actor.

There are two actors in our system:

1. Distributor

2. Agent

4.2.2 Sequence Diagram

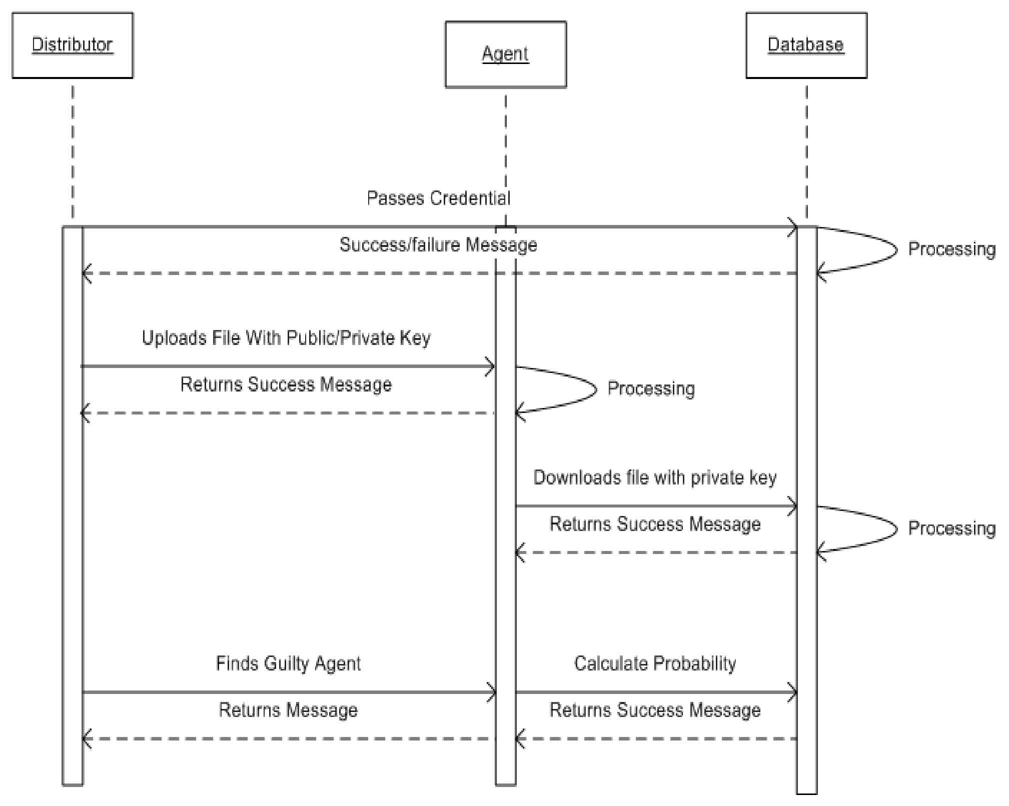


FIG 4.2.2: Sequence Diagram

Sequence diagram :

1. Sequence diagram shows the overall flow of the project.

2. The distributor logins into the system.

3. The message will be displayed showing the success or failure.

4. Agent asks for the particular file or distributor uploads all the files for the agents accordingly

along with the private key.

5. Agents will download the files according to his needs

6. If any agents leak the data to the third party the distributor will find out the probability of the particular agent who has leaked the file and will also find the file which has been leaked.

4.2.3 Activity Diagram

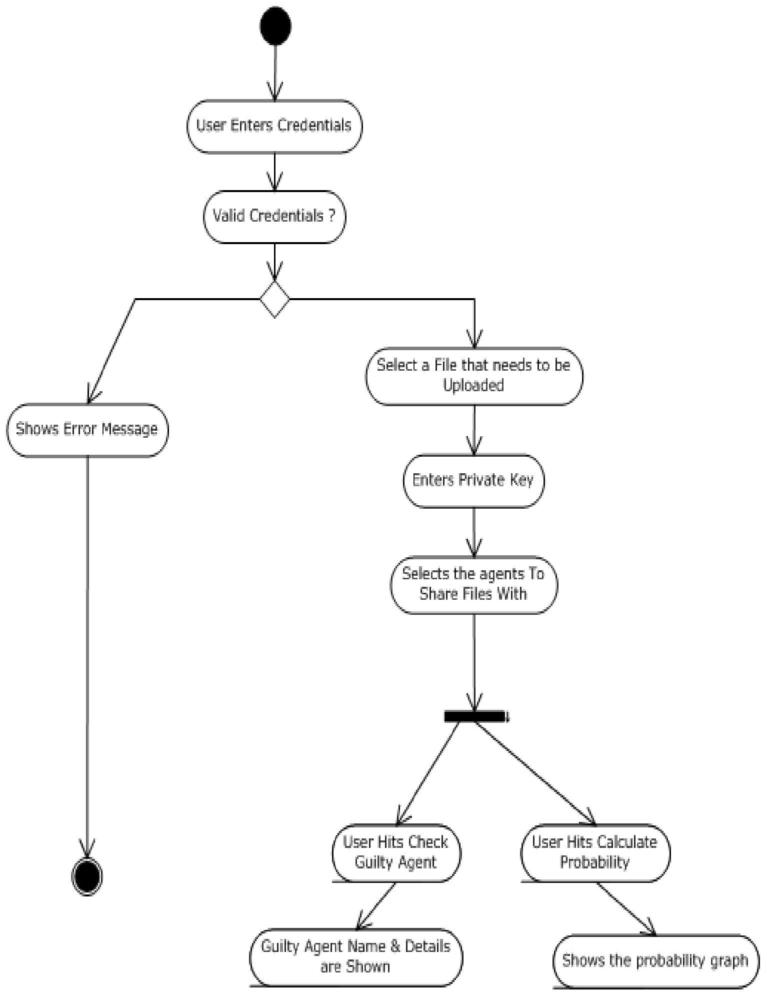

User Enters Credentials
Valid Credentials ?
Select a File that needs to be Uploaded
Shows Error Message
Enters Private Key
Selects the agents To Share Files With
User Hits Check Guilty Agent
User Hits Calculate Probability
Guilty Agent Name & Details are Shown
Shows the probability graph

FIG 4.4.3: Activity Diagram

4.2.4 Component Diagram

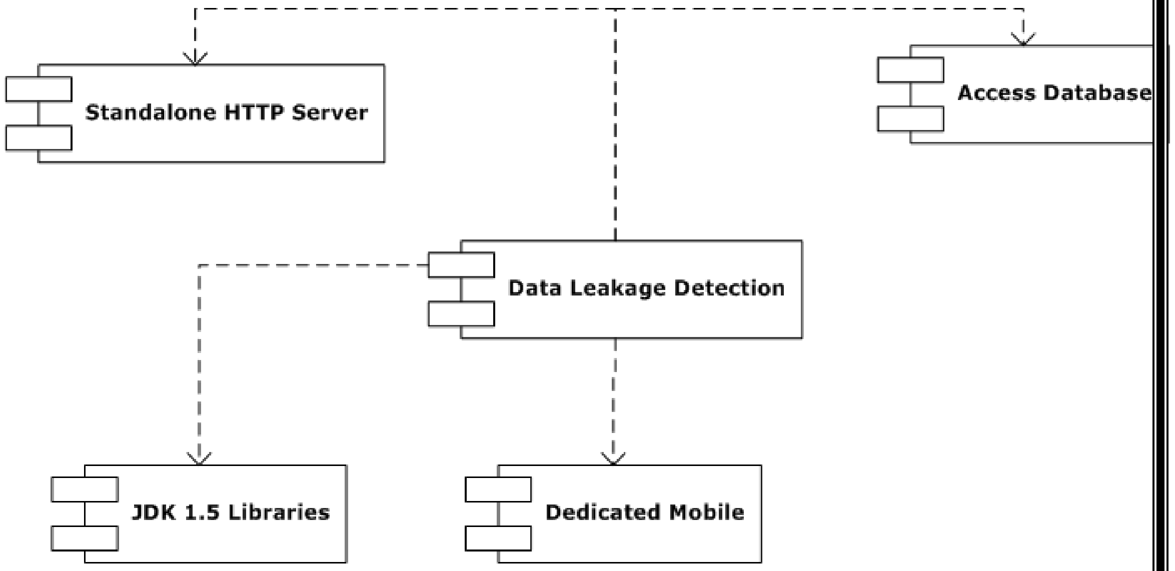


FIG 4.2.4: Component Diagram

The components used in our system are

Standalone HTTP Server

Access Database

Data Leakage System

JDK 1.5 Libraries

Dedicated Mobile (In case of failure of email sending)

4.2.5 Deployment Diagram

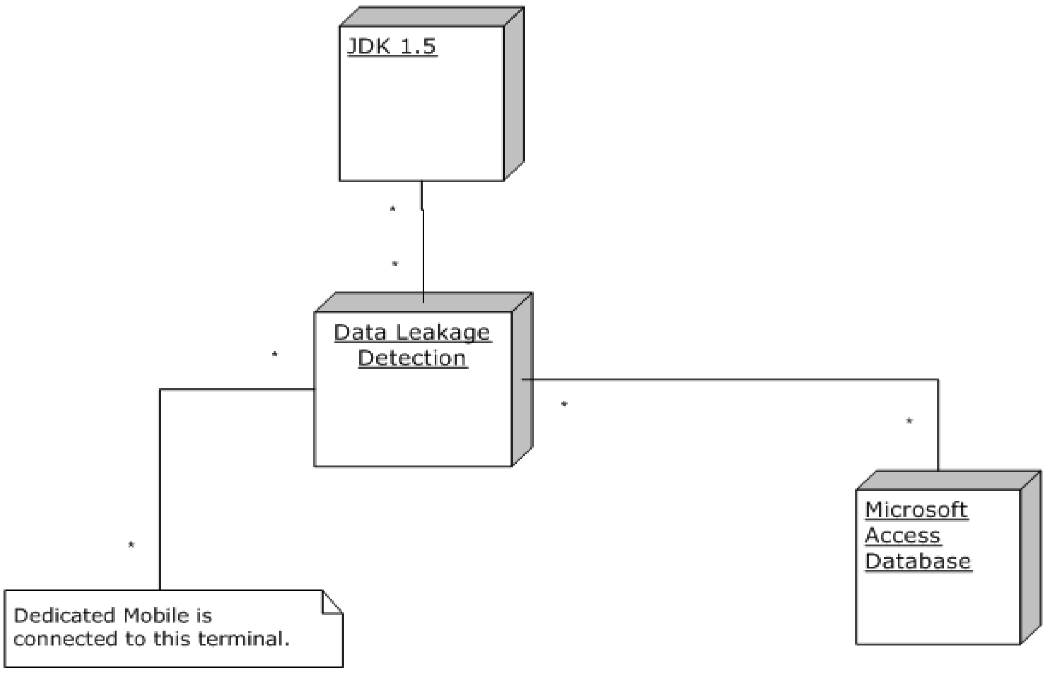

JDK 1.5
Data Leakage Detection
Microsoft Access Database
Dedicated Mobile is connected to this terminal.
*
*
*
*
*
*

FIG 4.2.5: Deployment Diagram

Deployment Diagram models the *physical* deployment of artifacts on nodes.

There are two types of Nodes

1. Device Node
2. Execution Environment Node

Device nodes are physically computing resources with processing memory and services to execute software, such as typical computer or mobile phones.EEN node is software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements.

The component Data Leakage Detection when executed, at that time all the components related to that component will come into play.

4.2.6 Work Flow Diagram

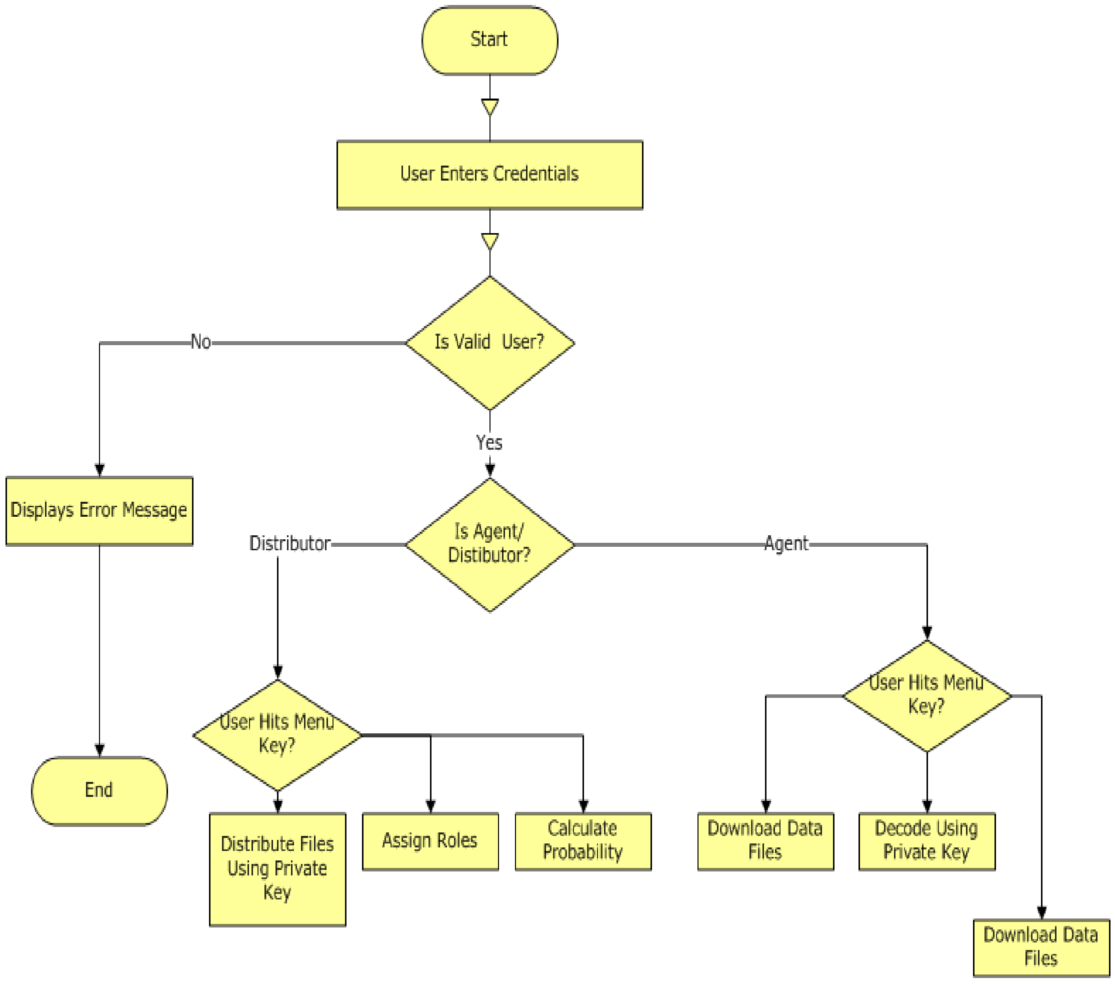


FIG 4.2.6: Work Flow Diagram

CHAPTER 5

TECHNICAL SPECIFICATIONS

5.1 TECHNOLOGY DETAILS USED IN PROJECT

5.1.1 MYSQL

MySQL is the world's most used relational database management system (RDBMS) that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack—LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python"

MySQL is primarily an RDBMS and ships with no GUI tools to administer MySQL databases or manage data contained within the databases. Users may use the included command line tools or download MySQL front-ends from various parties that have developed desktop software and web applications to manage MySQL databases, build database structures, and work with data records.

MySQL can be built and installed manually from source code, but this can be tedious so it is more commonly installed from a binary package unless special customizations are required. On most Linux distributions the package management system can download and install MySQL with minimal effort, though further configuration is often required to adjust security and optimization settings.

Though MySQL began as a low-end alternative to more powerful proprietary databases, it has gradually evolved to support higher-scale needs as well. It is still most commonly used in small to medium scale single-server deployments, either as a component in a LAMP-based web application or as a standalone database server. Much of MySQL's appeal originates in its relative simplicity and ease of use, which is enabled by an ecosystem of open source tools such as phpMyAdmin. In the medium range, MySQL can be scaled by deploying it on more powerful hardware, such as a multi-processor server with gigabytes of memory.

There are however limits to how far performance can scale on a single server, so on larger scales, multi-server MySQL deployments are required to provide improved performance and reliability. A typical high-end configuration can include a powerful master database which handles data write operations and is replicated to multiple slaves that handle all read operations. The master server synchronizes continually with its slaves so in the event of failure a slave can be promoted to become the new master, minimizing downtime. Further improvements in performance can be achieved by caching the results from database queries in memory using memcached, or breaking down a database into smaller chunks calledshards which can be spread across a number of distributed server clusters.

5.1.2 NETBEANS IDE

The NetBeans IDE is an award-winning integrated development environment available for Windows, Mac, Linux, and Solaris. The NetBeans project consists of anopen-source IDE and an application platform that enable developers to rapidly create web, enterprise, desktop, and mobile applications using the Java platform, as well as JavaFX, PHP, JavaScript and Ajax, Ruby and Ruby on Rails, Groovy and Grails, and C/C++.

The NetBeans Profiler is a tool for the monitoring of Java applications: It helps developers find memory leaks and optimize speed. Formerly downloaded separately, it is integrated into the core IDE since version 6.0.

The Profiler is based on a Sun Laboratories research project that was named JFluid. That research uncovered specific techniques that can be used to lower the overhead of profiling a Java application. One of those techniques is dynamic bytecode instrumentation, which is particularly useful for profiling large Java applications. Using dynamic bytecode instrumentation and additional algorithms, the NetBeans Profiler is able to obtain runtime information on applications that are too large or complex for other profilers. NetBeans also support Profiling Points that let you profile precise points of execution and measure execution time

Formerly known as *project Matisse*, the GUI design-tool enables developers to prototype and design Swing GUIs by dragging and positioning GUI components.

The NetBeans JavaScript editor provides extended support for JavaScript, Ajax, and CSS.

JavaScript editor features comprise syntax highlighting, refactoring, code completion for native objects and functions, generation of JavaScript class skeletons, generation of Ajax callbacks from a template; and automatic browser compatibility checks..

5.1.3 JAVA.COMM API

The Java Communications 3.0 API is a Java extension that facilitates developing platform-independent communications applications for technologies such as Smart Cards, embedded systems, and point-of-sale devices, financial services devices, fax, modems, display terminals, and robotic equipment.

The Java Communications API (also known as javax.comm) provides applications access to RS-232 hardware (serial ports) and limited access to IEEE-1284 (parallel ports), SPP mode.

Implementations of the API are currently available for Solaris SPARC, Solaris x86, and Linux x86. Each of Sun's available implementations works with the **Sun Ray** thin client product line, and include portmapping extensions to allow an administer to specify the locations of ports as well as their visibility, names, and in some cases annotated reference.

API serial features:

- Enumeration of ports (administrator and user configurable port mapping)
- Port configuration (baud rate, speed, stop bits, parity)
- Access to EIA232 standard DTR, CD, CTS, RTS and DSR signals
- Transfer of data over RS-232 ports
- Hardware and software flow-control options
- Receive-buffer threshold control
- Asynchronous event option for notification of:
  - Data available on an RS-232 port
  - Port hardware line level changes

- Port ownership changes within a single JVM

5.1.4 XAMPP SERVER

XAAMP is a free and open source cross-platform web server solution stack package, consisting mainly of the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages.

Installing XAMPP takes less time than installing each of its components separately. Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another.

It is offered in both a full, standard version and a smaller version.

Officially, XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the Internet. To make this as easy as possible, many important security features are disabled by default. In practice, however, XAMPP is sometimes used to actually serve web pages on the World Wide Web. A special tool is provided to password-protect the most important parts of the package

XAMPP also provides support for creating and manipulating databases in MySQL and SQLite among others.

CHAPTER 6

PROJECT ESTIMATE

6.1 ESTIMATION BASED FUNCTION POINT

Q NO	QUESTION	WEIGHT FACTOR (0-5)
1	Does the System require reliable backup & Recovery?	2
2	Are data communications required to transfer information to or from the application?	3
3	Are there distributed processing functions?	0
4	Is performance critical?	4
5	Will the system run an existing, heavily utilized operational environment?	2
6	Does online data entry requires input transaction to be built on multiple screens?	4
7	Does system requires online data entry?	2
8	Are master files updated online?	3
9	Are i/p,o/p files complex?	2
10	Is internal processing complex?	4
11	Is code reusable?	2
12	Are conversions & installation included in design?	3

13	Does system support multiple installations?	1
14	Is application designed to facilitate change & ease of use by user?	2
	Total(Fi)	34

Table 6.1 Project Estimation

6.2 SCHEDULE:

Time schedule is very important. The project should be completed within fixed schedule as far as company is concerned. Beside this project is assigned to student as academic exercise as to be completed within fixed Schedule time. The duration required to complete “M-Control” is around 6 months. The table below holds the time required to complete each phase of our project:

• ANALYSIS	15 DAYS
• DESIGN OF SOFTWARE	2 MONTHS
• CODING	2 ½ MONTHS
• TESTING	15 DAYS
• POST MAINTENANCE	15 DAYS

Table 6.2 Schedule

CHAPTER 7

SOFTWARE IMPLEMENTATION

7.1 IMPORTANT MODULE AND ALGORITHMS

Key Operation in Application

1) Distributor sends file
2) Agent can download file
3) Agent can request
4) Distributor finds the probability
5) Admin or distributor can block the guilty agent

Algorithms

1)Sample Request algorithm

Input:- m1,m2…….mN ,|T| Assuming mi<=|T|

Output:- R1,R2…..Rn

```
1: a ← 0|T|                a[k]:number of agents who have received object tk
2: R1 ← ∅, . . . , Rn ← ∅
3: remaining ← $\sum_{i=1}^{n} m_i$
4: while remaining > 0 do
5: for all i = 1, . . . , n : |Ri| < mi do
```

```
6: k ← SELECTOBJECT(i,Ri)
```

May also use additional Parameters

```
7: Ri ← Ri ∪ {tk}
8: a[k] ← a[k] + 1
9: remaining ← remaining − 1
```

Thus we get the set of data objects (R1,R2….Rn) to be send to the particular Agent.(U1,U2…..Un)

In lines 1 and 2 of Algorithm 4, the main loop in lines 4-9 is executed while there are still data objects (remaining > 0) to be allocated to agents.

In each iteration of this loop (lines 5-9), the algorithm uses function SELECTOBJECT() to find a random object to allocate to agent Ui. This loop iterates over all agents who have not received the number of data objects they have requested.

(2) Explicit Request Algorithm :

Input:-R1, . . .,Rn ;cond1,. . . , condn ; b1, . . . , bn, B

Output:- R1, . . .,Rn ; F1, . . . ,Fn

```
1: R ← ∅ Agents that can receive fake objects
2: for i = 1, . . . , n do
3: if bi > 0 then
4: R ← R ∪ {i}
5: Fi ← ∅
6: while B > 0 do
7: i ← SELECTAGENT(R,R1, . . . , Rn)
8: f ← CREATEFAKEOBJECT(Ri, Fi, condi)
9: Ri ← Ri ∪ {f}
10: Fi ← Fi ∪ {f}
11: bi ← bi − 1
12: if bi = 0 then
```

```
13: R ← R\{Ri}
14: B ← B − 1
```

Thus we get the record set(R1,R2….Rn) for the particular condition(cond1….condn) which has been given by the agents(U1,U2….Un)

In lines 1-5, Algorithm 1 finds agents that are eligible to receiving fake objects in O(n) time. Then, in the main loop in lines 6-14, the algorithm creates one fake object in every iteration and allocates it to random agent .The main loop takes O(B) time. Hence, the running time of the algorithm is O(n + B).

(3)Encyption Algorithm (AES Algorithm)

1. KeyExpansion—round keys are derived from the cipher key using Rijndael's key schedule
2. Initial Round
   1. AddRoundKey—each byte of the state is combined with the round key using bitwise xor
3. Rounds
   1. SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
   2. ShiftRows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
   3. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
   4. AddRoundKey
4. Final Round (no MixColumns)
   1. SubBytes
   2. ShiftRows
   3. AddRoundKey

CHAPTER 8

SOFTWARE TESTING

8.1 INTRODUCTION

Testing is a process used to uncover errors and ensure that defined input will produce actual results that agree with required results. A strategy for software testing integrates software test case design methods into a well planned series of steps that result in successful construction of software.

The strategy for our project is as follows:

- Testing will begin at the component level and will work outward towards the integration of the entire system.
- Appropriate testing techniques will be used at different points in time.
- Debugging is an activity that will go hand in hand with testing.

8.2 TEST CASES

LOGIN PAGE TEST CASES:

SR.NO.	STEP DESCRYPTION	EXPECTED RESULTS
1	Go to User Id field and without entring data in that field press "Enter" key.	it should prompt message " Please enter 'User ID' "
2)	Go to login screen enter "User Id" and without entering Password tries to click on "OK" button.	it should prompt message "Please enter 'Password'".

3)	Go to login screen enter "User Id" and enter wrong Password tries to click on "OK" button.	it should prompt message " Please enter Proper 'User ID' and 'Password'".
4)	Go to Login screen enter all required data and press "OK" Button.	Software window will open.

Table 8.2.1 Login Page Test Cases

REGISTRATION FORM TEST CASES:

SR.NO.	STEP DESCRYPTION	EXPECTED RESULTS
1	enter first name and last name blank	it should prompt message "first name and last name should not be blank”
2)	Enter the contact no less than 10 digit	it should prompt message "enter 10 digit contact no”
3)	Enter not valid email id	it should prompt message "enter valid email id ”
4)	Enter already exist login id and password	it should prompt message “login id is in use”.

5)	If password and conform password field not matched.	it should prompt message “conform password not matched”.

Table 8.2.2 Registration Form Test Cases

AGENT PAGE TEST CASES:

SR.NO.	STEP DESCRYPTION	EXPECTED RESULTS
1	If agent click on change password	It should open password change window
2)	If agent enter new password correctly.	it should prompt message "password change successfully”.
3)	If agent click on download files.	It should open download file window.
4)	At the time of download files if user enter wrong Encryption key.	it should prompt message “enter correct Encryption key”.
5)	At the time of download files if user enters correct Encryption key.	File contents should be display on text box area.
6)	If agent not select any file and click on download button	It should prompt message “you have no select any file”
7)	If agent click on send request	It should open request window
8)	If agent not select any distributor at the time of sending request	It should prompt message “you have not select any distributor “.
9)	If agent click on Log out	It should go to login page.

Table 8.2.3 Agent Page Test Cases

DISTRIBUTOR PAGE TEST CASES:

SR.NO.	STEP DESCRYPTION	EXPECTED RESULTS
1	If distributor click on change password	It should open password change window
2)	If distributor enter new password correctly.	it should prompt message "password change successfully”.
3)	If distributor click on upload the file	It should open upload window.
4)	If distributor not select agent	It should prompt message “ select agent”
	If distributor not enter Encryption key.	It should prompt message “ Enter Encryption key ”
4)	On click on share button	File should be share with no error and shared file details should be display below.
5)	If distributor click on find probability	It should open probability window.
6)	If distributor click on Log out	It should go to login page.

Table 8.2.4 Distributor Form Test Cases

ADMINISTATOR PAGE TEST CASES:

SR.NO.	STEP DESCRYPTION	EXPECTED RESULTS
1	If administrator click on change password	It should open password change window
2)	If administrator enter new password correctly.	it should prompt message "password change successfully”.
3)	If administrator click on upload the file	It should open upload window.
4)	If administrator not select agent	It should prompt message “ select agent”
	If administrator not enter Encryption key.	It should prompt message “ Enter Encryption key ”
4)	On click on share button	File should be share with no error and shared file details should be display below.
5)	If administrator click on find probability	It should open probability window.
6)	If administrator click on block guilty agent	It should open block guilty agent window
7)	If administrator select guilty agent and enter description properly	It should send email on that particular agent’s id.
8)	If administrator click on Log out	It should go to login page.

Table 8.2.5 Administrator Form Test Cases

CHAPTER 9

RESULTS (SNAPSHOTS OF RESULTS)

1) Administrator

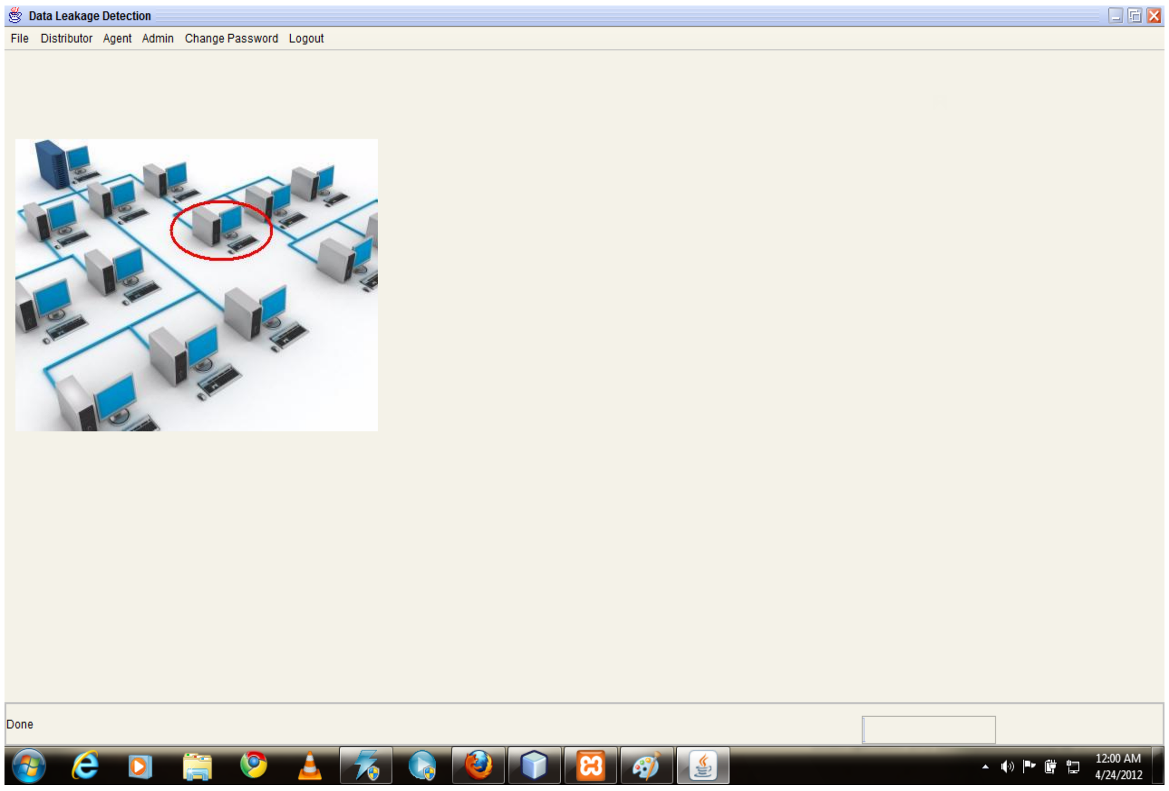
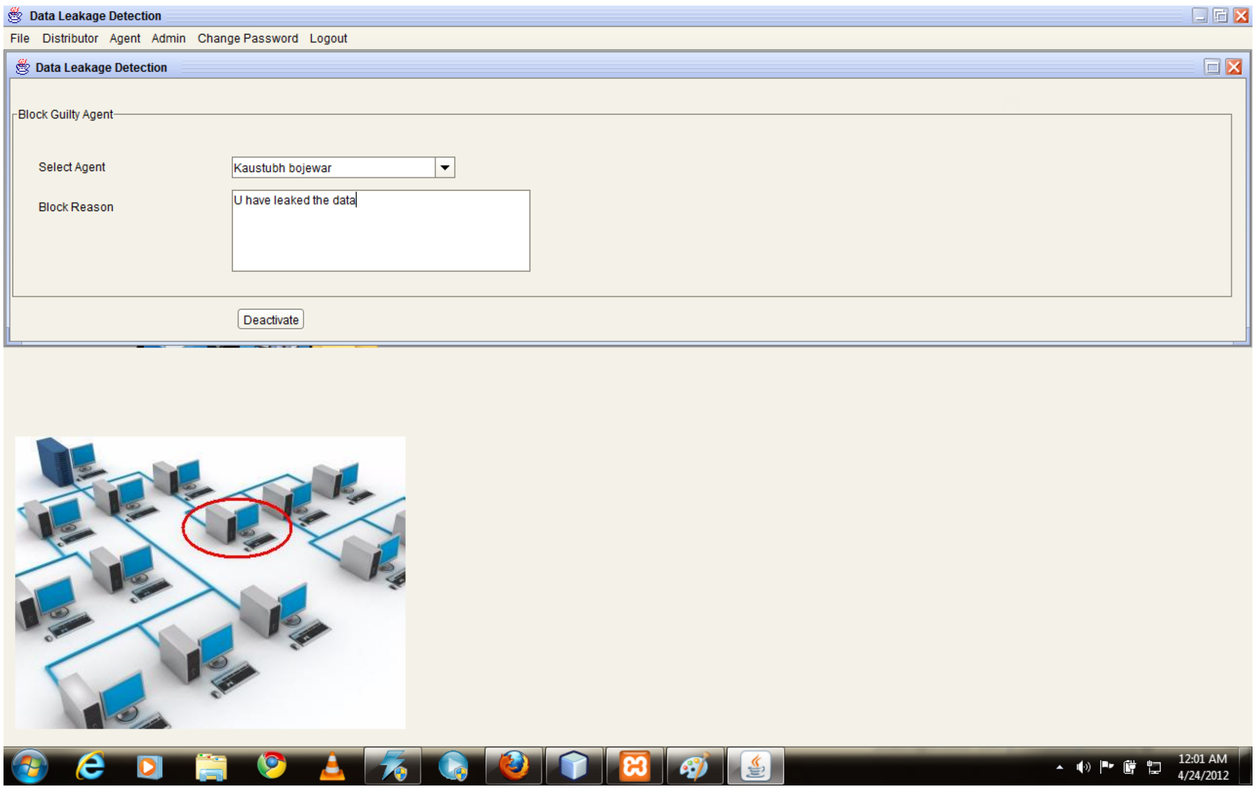


FIG 9.1: Administrator form

This form will be visible to Administrator.As he have the right to view everything going on in the application,every menu including file,distributor,agent,admin,change password will be visible to him.He can see the probability distribution table n accordingly block the guilty agent.

2) Administrator blocking Agent :



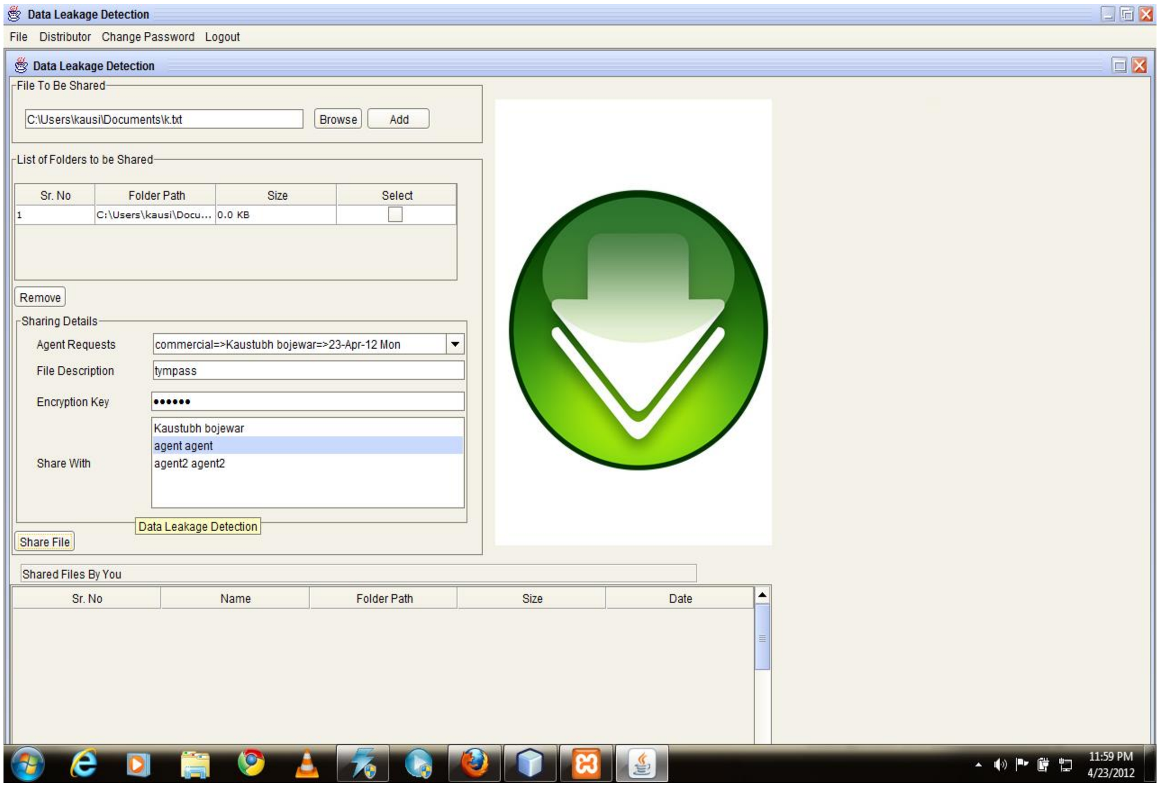
Block Guilty Agent

Select Agent	Kaustubh bojewar
Block Reason	U have leaked the data

FIG 9.2:Administrator blocking agent form

This form shows how the Administrator blocks the agent who has leaked the data.He will first select the agent from the drop down list of the agents and in the next text box he will give a reason why he had block the agent.

3) Distributor Upload File



File To Be Shared: C:\Users\kausi\Documents\k.txt

List of Folders to be Shared

Sr. No	Folder Path	Size	Select
1	C:\Users\kausi\Docu...	0.0 KB	☐

Sharing Details

Agent Requests	commercial=>Kaustubh bojewar=>23-Apr-12 Mon
File Description	tympass
Encryption Key	••••••
Share With	Kaustubh bojewar agent agent agent2 agent2

Shared Files By You

Sr. No	Name	Folder Path	Size	Date

FIG 9.3: Distributor form

This is one of the main forms in our project.This form is for the distributor.In both the main modules distributor distributes the files to the agent. In the first algorithm the distributor decides which data is to be sent and in the second algo the agents sends the request to the distributor and accordingly distributor sends the data to the particular agent through this form.

4) Agent

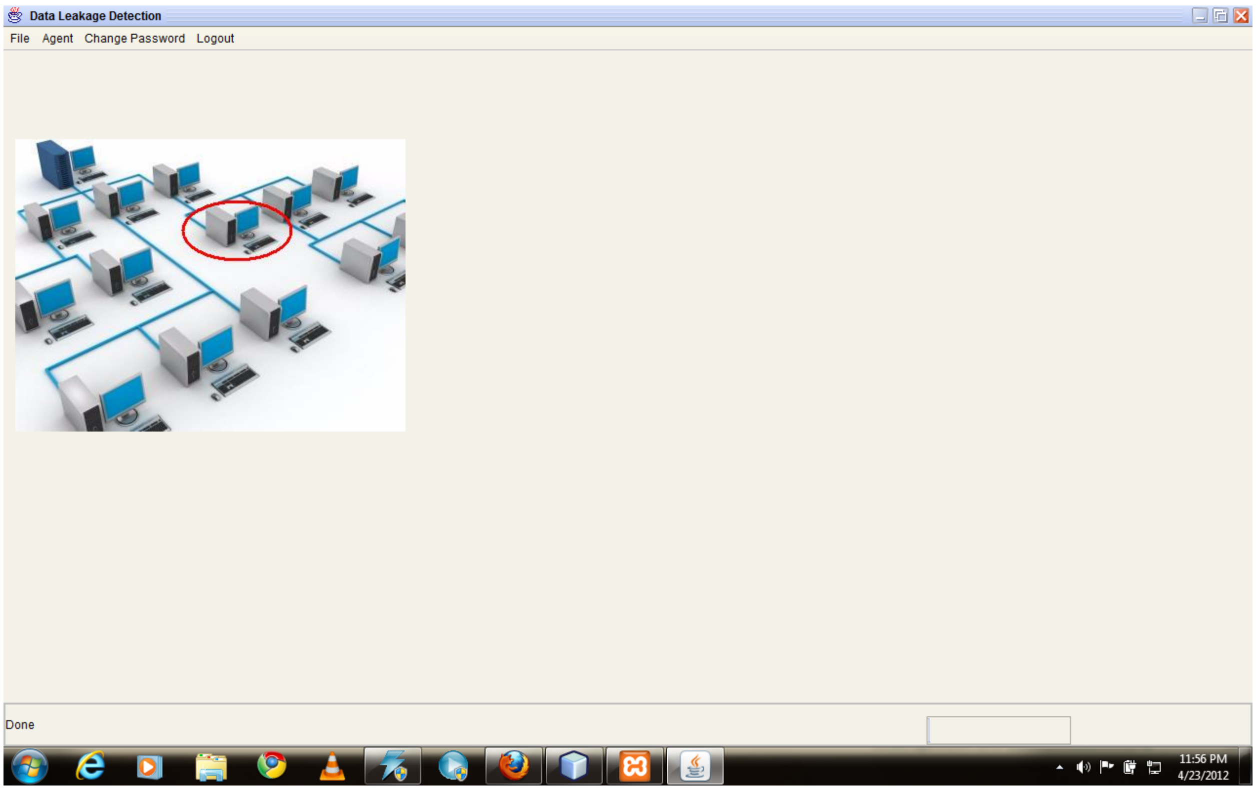
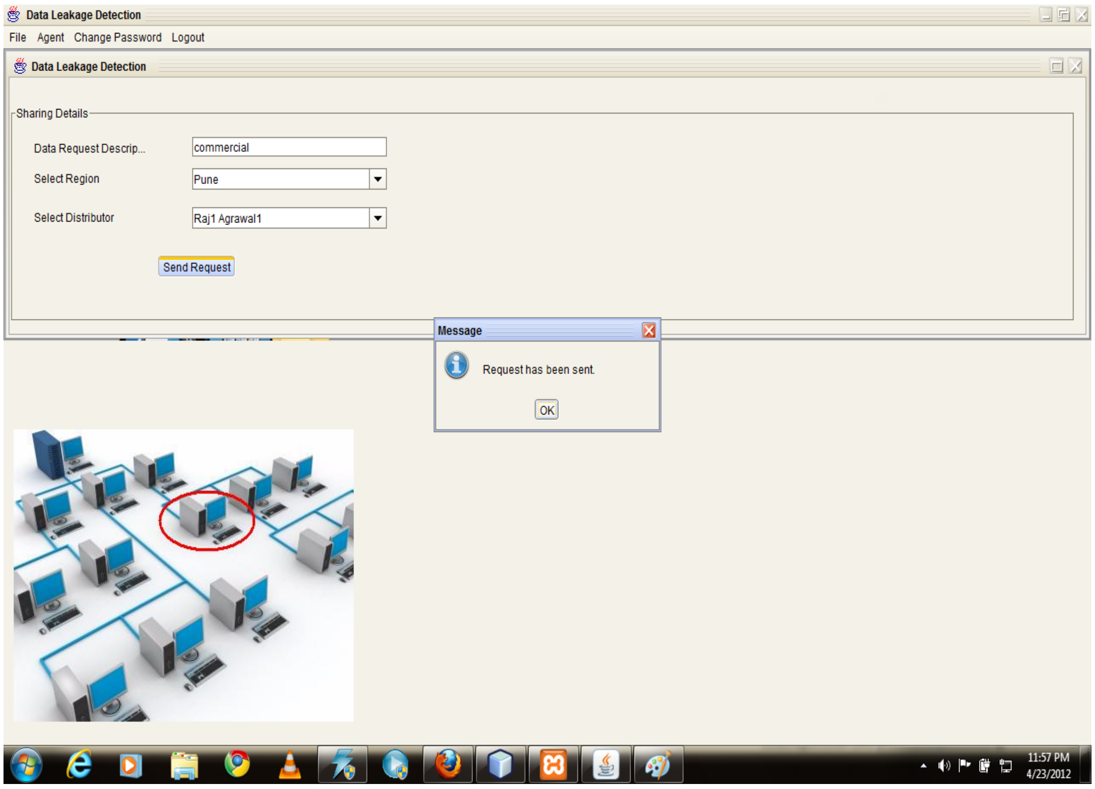


FIG 9.4:Agent form

This is the Agents's form.Here the agent can download the files which distributor has sent.Wheather using any of the algorithms.The agent cant download the file on the machine,he can only view the data on the aaplication.

5) Agent Send Request



Sharing Details

Data Request Descrip...	commercial
Select Region	Pune
Select Distributor	Raj1 Agrawal1

Request has been sent.

FIG 9.5:Agent send request

One more thing that agent can do is he can send the request to the distributor asking for the particular type of files.In that case this form is used.Where he has to provide a request and send it to distributor.

CHAPTER 10

DEPLOYMENT AND MAINTENANCE

10.1 INSTALLATION AND UNINSTALLATION

INSTALLATION

1. Need to install Netbeans
2. Need to install JDK on both side client and server side
3. Need to install MySql on server side
4. Need to install XAMPP
5. For Client Side: make exe of project and run on the client side.

UNINSTALLATION

1. Need to uninstall Netbeans from control panel
2. Need to uninstall MySql from control panel

10.2 USER HELP

AGENT:

1. Open the Application
2. Enter agent id and password
3. Agent can do 2 operations, Request for file or Download file
4. For download select particular file and ask key from distributor(call distributor for encryption key)
5. Enter the key
6. File contents will display on screen

DISTRIBUTOR:

1.Open the Application

2. Enter distributor id and password

3. Distributor can do 2 operations, send the file, find the guilty agent

4. For sending file : distributor have to select particular File and select agents whom to send

5 For finding Guilty agent: click on calculate guilty agent and find guilty agent

ADMINISTRATOR:

1. Open the Application

2. Enter Administrator id and password

3. Distributor can do 3 operations, send the file, find the guilty agent, block the guilty agent

4. For sending file : distributor have to select particular File and select agents whom to send

5 For finding Guilty agent: click on calculate guilty agent and find guilty agent

6 For Blocking Guilty agent: send some message to agent and block id and password.

CONCLUSION AND FUTURE SCOPE

Conclusion

In a perfect world there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to hand over sensitive data, in a perfect world we could watermark each object so that we could trace its origins with absolute certainty. However, in many cases we must indeed work with agents that may not be 100% trusted, and we may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks. In spite of these difficulties, we have shown it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means. Our model is relatively simple, but we believe it captures the essential trade-offs. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. We have shown that distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive.

It includes the investigation of agent guilt models that capture leakage scenarios that are not studied in this paper. For example, what is the appropriate model for cases where agents can collude and identify fake tuples? A preliminary discussion of such a model is available in. Another open problem is the extension of our allocation strategies so that they can handle agent requests in an online fashion (the presented strategies assume that there is a fixed set of agents with requests known in advance).

Future Scope

The developer of an application can never be carried out to the fullest extend in a stipulated time, the main reason why revisions of the application are always introduced in course of time. This application being restricted to one time development will have no revision done, hence certain areas that can be enhanced is pointed.

- The enhanced version of this system can detect the guilty agent even through the internet.

REFERENCES

[1] "Data Leakage Detection"Panagiotis Papadimitriou, Student Member, IEEE, and Hector Garcia-Molina, Member, IEEE

[2] R. Agrawal and J. Kiernan, "Watermarking Relational Databases," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), VLDB Endowment, pp. 155-166, 2002.

[3] P. Bonatti, S.D.C. di Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," ACM Trans. Information and System Security, vol. 5, no. 1, pp. 1-35, 2002.

[4] P. Buneman, S. Khanna, and W.C. Tan, "Why and Where: A Characterization of Data Provenance," Proc. Eighth Int'l Conf. Database Theory (ICDT '01), J.V. den Bussche and V. Vianu, eds., pp. 316-330, Jan. 2001

[5] P. Buneman and W.-C. Tan, "Provenance in Databases," Proc. ACM SIGMOD, pp. 1171-1173, 2007.

[6] Y. Cui and J. Widom, "Lineage Tracing for General Data Warehouse Transformations," The VLDB J., vol. 12, pp. 41-58, 2003.

[7] F. Hartung and B. Girod, "Watermarking of Uncompressed and Compressed Video," Signal Processing, vol. 66, no. 3, pp. 283-301,

1998.

[8] S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian,"Flexible Support for Multiple Access Control Policies," ACM

Trans. Database Systems, vol. 26, no. 2, pp. 214-260, 2001.

[9] Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting RelationalDatabases: Schemes and Specialties," IEEE Trans. Dependable and

Secure Computing, vol. 2, no. 1, pp. 34-45, Jan.-Mar. 2005.

[10] B. Mungamuru and H. Garcia-Molina, "Privacy, Preservation and Performance: The 3 P's of Distributed Data Management,"

technical report, Stanford Univ., 2008.

[11] V.N. Murty, “Counting the Integer Solutions of a Linear Equation with Unit Coefficients,” Math. Magazine, vol. 54, no. 2, pp. 79-81,

1981.

[12] A programming guide to java certification

[13] Cryptograpphy and network security by Williams Stallings

[14] Java Complete Reference and Black Book by Kogent Solutions

APPENDIX A

GLOSSARY

Watermarking

A **watermark** is a recognizable image or pattern in paper that appears as various shades of lightness/darkness when viewed by transmitted light (or when viewed by reflected light, atop a dark background), caused by thickness or density variations in the paper.

Encryption

It is the process of converting plain text into the ciphertext which is not readable in general form using the encpyption algorithm for the security purpose.

Class diagrams:

Class diagrams use classes and interfaces to capture details about the entities that make up your system and the static relationships between them. Class diagrams are one of the most commonly used UML diagrams, and they vary in detail from fully fleshed-out and able to generate source code to quick sketches on whiteboards and napkins.

Component diagrams:

Component diagrams show the organization and dependencies involved in the implementation of a system. They can group smaller elements, such as classes, into larger, deployable pieces. How much detail you use in component diagrams varies depending on what you are trying to show. Some people simply show the final, deployable version of a system, and others show what functionality is provided by a particular component and how it realizes its functionality internally.

Deployment diagrams:

Deployment diagrams show how your system is actually executed and assigned to various pieces of hardware. You typically use deployment diagrams to show how components are configured at runtime.

Activity diagrams:

Activity diagrams capture the flow from one behavior or *activity*, to the next. They are similar in concept to a classic flowchart, but are much more expressive.

Sequence diagrams:

Sequence diagrams are a type of interaction diagram that emphasize the type and order of messages passed between elements during execution. Sequence diagrams are the most common type of interaction diagram and are very intuitive to new users of UML.

Use case diagrams:

Use case diagrams capture functional requirements for a system. They provide an implementation-independent view of what a system is supposed to do and allow the modeler to focus on user needs rather than realization details.