**ABSTRACT**

When the internet is available to all users the issue was held on security even in the present situation the security is major issue because the information which we transmit isn’t p2p so the information may even seen by the third party so the research have been started on it and invented some methods behalf of it those are cryptography, steganography and more. Here the cryptography is the technique in which the message is converted to cipher-text so that the message is seen but readable format and the other one is steganography in which the hiding of information takes place so that it cannot be seen even. There are many technique to implement on steganography they are hiding information in image, hiding information in video/audio (any media), and at last in text. This project explores one of the methods of steganography called text steganography approach by considering differently spelt words of different languages. In this method the US and UK spellings of words substituted in order to hide data in an English text. For example "color" has different spelling in UK (colour) and US (color). Therefore the data can be hidden in the text by substituting these words.

**CONTENTS**

|  |  |
| --- | --- |
|  |  |
| **LIST OF FIGURES** | IV |
| **LIST OF TABLES** | V |
| **LIST OF SCREENS** | VI |
| **LIST OF ACRONYMS** | VII |
| **CHAPTER 1** |  |
| **INTRODUCTION** |  |
| 1.1 Motivation | 1 |
| 1.2 Problem Definition | 1 |
| 1.3 Objective of Project | 1 |
| 1.4 Organization of Project | 2-3 |
| **CHAPTER 2** |  |
| **LITERATURE SURVEY** |  |
| 2.1 Introduction | 4 |
| 2.2 Existing Systems | 4-5 |
| 2.3 Disadvantages of Existing System | 6 |
| 2.4 Proposed System | 7 |
| **CHAPTER 3**  **REQUIREMENT SPECIFICATIONS**  3.1 Software requirement 8  3.2 Hardware requirement 9  **CHAPTER 4**  **MODULE DESCRIPTION**  4.1 Introduction 10-12  4.2Admin Module 13  4.3 User Module 14  **CHAPTER 5** |  |
| **DESIGN AND ANALYSIS** |  |
| 5.1 System Architecture | 15 |
|  |  |
| 5.2 Entity Relationship Diagram 16-17 |  |
| 5.2 UML diagram (any other project diagrams) 18-30 |  |
| **CHAPTER 6** |  |
| **IMPLEMENTATION AND RESULTS** |  |
| 6.1 Introduction | 30-34 |
| 6.2 Source Code | 35-36 |
| 6.3 Implementation | 37 |
| 6.3.1 ScreenShots | 38-43 |
| 6.3.2 Database connectivity | 44-47 |
| 6.3.3 Result Analysis | 48 |
| **CHAPTER 7** |  |
| **TESTING & VALIDATION** |  |
| 7.1 Introduction | 49 |
| 7.2 Design of test cases and scenarios | 50 |
| **CHAPTER 8** |  |
| **CONCLUSION** |  |
| 8.1 Future Enhancement | 51 |
| 8.2 Project Conclusion | 51 |
| **REFERENCES** | 52 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Fig 5.1 | System Architecture | 15 |
| Fig 5.2 | Entity Relationship diagram | 17 |
| Fig 5.3.1.1 | Class diagram | 20 |
| Fig 5.3.1.2 | Component Diagram | 21 |
| Fig 5.3.1.3 | Use case Diagram | 22 |
| Fig 5.3.1.4 | Sequence Diagram | 24 |
| Fig 5.3.1.5(a) | State chart Diagram for user | 25 |
| Fig 5.3.1.5(b) | State chart Diagram for admin | 26 |
| Fig 5.3.1.6(a) | Activity Diagram for user | 27 |
| Fig 5.3.1.6(b) | Activity Diagram for admin | 28 |
| Fig 5.3.1.7 | Deployment Diagram | 29 |
| Fig 5.3.1.8 | Collaboration Diagram | 30 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF TABLES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table 7.1.1 | List of Test Cases | 50 | Name of the Table | 7 |

**LIST OF SCREENS**

|  |  |  |
| --- | --- | --- |
| Screen 6.3.1 | Home page | 37 |
| Screen 6.3.2 | User Registration page | 38 |
| Screen 6.3.3 | User login | 38 |
| Screen 6.3.4 | Category selection by user | 39 |
| Screen 6.3.5 | User views questions | 39 |
| Screen 6.3.6 | User select the question and answers | 40 |
| Screen 6.3.7 | View result | 40 |
| Screen 6.3.8 | View tips | 41 |
| Screen 6.3.9 | Tips and logout selection | 41 |
| Screen 6.3.10 | Tips | 42 |
| Screen 6.3.11 | Admin login | 42 |
| Screen 6.3.12 | Admin chooses category | 43 |

**LIST OF ACRONYMS**

|  |  |
| --- | --- |
| ASCII | American Standard Code for Information Interchange |
| STEGO | Steganography |
| AES | Advance Encryption Standard |
| GB | Giga Bytes |
| JRE | Java Runtime Environment |
| JDK | Java Development Kit |
| ERD | Entity-Relationship Diagram |
| UML | Unified Modeling Language |
| IDE | Integrated Development Environment |

**INTRODUCTION**

**1.1 Introduction**

The world is becoming more interconnected with the advent of the Internet and new networking technology. There is a large amount of personal, commercial, military, and government information on networking infrastructures worldwide. Network security is becoming a great importance because of intellectual property that can be easily acquired through the internet [1].

There are currently two fundamentally different networks, data networks and synchronous network comprised of switches. The internet is considered a data network, since the current data network consists of computer‐based routers, information can be obtained by special programs such as “Trojan horses” planted in the routers. The synchronous network that consists of switches does not buffer data and therefore are not threatened by attackers. That is why security is emphasized in data networks, such as the internet, and other networks that link to the internet [2].

When considering network security, it must be emphasized that the whole network is secure. Network security does not only concern the security in the computers at each end of the communication chain. When transmitting data the communication channel should not be vulnerable to attack. A possible hacker could target the communication channel, obtain the data, and decrypt it and re‐insert a false message. Securing the network is just as important as securing the computers and encrypting the message.

When developing a secure network the following things need to be considered [3]:

1. **Access**-Authorized users are provided the means to communicate to and from a particular network.
2. **Confidentiality**-Information in the network remains private.
3. **Authentication** – Ensure the users of the network are who they say they are.
4. **Integrity** – Ensure the message has not been modified in transit.
5. **Non‐repudiation** – Ensure the user does not refute that he used the network.

As stated in the figure 1 for a secure transmission of data from one corner of the world to another the most commonly used algorithms are as follows

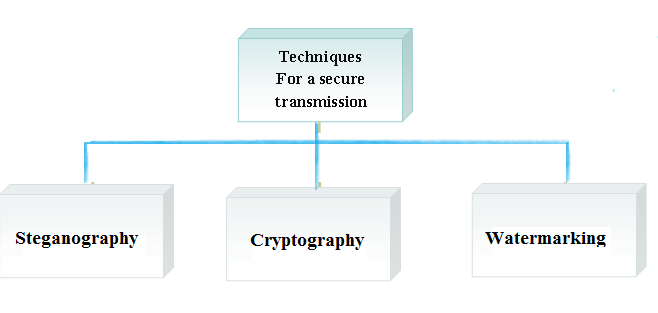


Figure 1 Secure transmission techniques

1. Cryptography
2. Steganography
3. Watermarking
   1. **Overview of Cryptography**

Cryptography is the study of hiding information and it is used when communicating over an un-trusted medium such as internet, where information needs to be protected from other third parties [4]. Modern cryptography focuses on developing cryptographic algorithms that are hard to break by an adversary due to the computational hardness therefore could not be broken by a practical means. In the modern cryptography, there are three types of cryptographic algorithms used [5]:

1. Symmetric key cryptography.
2. Public key cryptography.
3. Hash Functions.

Symmetric key cryptography involves encryption methods where both the sender and the receiver share the same key used to encrypt the data. In Public-key cryptography, two different but mathematically related keys are used. Hash functions does not use a key, instead they compute a fixed length hash value from the data. It is impossible to recover the length or the original plain text from this hash value.

* 1. **Overview of Steganography**

Steganography deals with composing hidden messages so that only the sender and the receiver know that the message even exists. Since nobody except the sender and the receiver knows the existence of the message, it does not attract unwanted attention. Steganography was used even in ancient times and these ancient methods are called Physical Steganography. Some examples for these methods are messages hidden in messages body, messages written in secret inks, messages written on envelopes in areas covered by stamps, etc [6].

Modern Steganography methods are called Digital Steganography. These modern methods include hiding messages within noisy images, embedding a message within random data, embedding pictures with the message within video files, etc.

Furthermore, Network Steganography is used in telecommunication networks. This includes techniques like Steganophony (hiding a message in Voice-over-IP conversations) and WLAN Steganography (methods for transmitting Steganograms in Wireless Local Area Networks).

* 1. **Overview of Watermarking**

Watermarking is the process of embedding a message on a host signal.  Watermarking, as opposed to Steganography, has the additional requirement of robustness against possible attacks.

Watermarking is of two types

1. visible watermarking
2. Invisible watermarking.

**Visible Watermarking**, as the name suggests, visible watermarking refers to the information visible on the image or video or picture. Visible watermarks are typically logos or text. For example, in a TV broadcast, the logo of the broadcaster is visible at the right side of the screen.

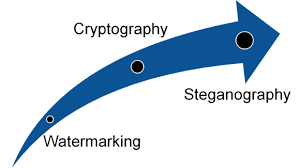
**Invisible Watermarking,** Invisible watermarking refers to adding information in a video or picture or audio as digital data. It is not visible or perceivable, but it can be detected by different means. It may also be a form or type of Steganography and is used for widespread use. It can be retrieved easily.

Using digital watermarking, copyright information can be embedded into the multimedia data. This is done by using some algorithms. Information such the serial number, images or text with special significance can be embedded. The function of this information can be for copyright protection, secret communication etc.

* 1. **Difference between Cryptography and Steganography**

Cryptography is the study of hiding information, while Steganography deals with composing hidden messages so that only the sender and the receiver know that the message even exists. In Steganography, only the sender and the receiver know the existence of the message, whereas in cryptography the existence of the encrypted message is visible to the world. Due to this, Steganography removes the unwanted attention coming to the hidden message. Cryptographic methods try to protect the content of a message, while Steganography uses methods that would hide both the message as well as the content. The research on Steganography has rapidly increased in recent years when compared to cryptography [3].

Steganographic technologies are a very important part of the future of Internet security and privacy on open systems such as the Internet. Even the research has been increased for steganography when compared to cryptography fig 2. Steganographic research is primarily driven by the lack of strength in the cryptographic systems on their own and the desire to have complete secrecy in an open-systems environment. Many governments have created laws that either limit the strength of cryptosystems or prohibit them completely. This has been done primarily for fear by law enforcement not to be able to gain intelligence by wiretaps, etc. This unfortunately leaves the majority of the Internet community either with relatively weak and a lot of the times breakable encryption algorithms or none at all. Civil liberties advocates fight this with the argument that “these limitations are an assault on privacy”. This is where Steganography comes in. Steganography can be used to hide important data inside another file so that only the parties intended to get the message even knows a secret message exists.



**Figure 2 Watermarking to Cryptography to Steganography**

The following tabular form shows the major differences between cryptography and Steganography[5]

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CRYPTOGRAPHY** | **STEGANOGRAPHY** |
| 1 | Known message is passing is done | Unknown Message passing is done |
| 2 | Encryption prevents an unauthorized party from recovering the contents from communication | Steganography prevents discovery of the very existence of communication |
| 3 | Common technology is used | Little known technology is used |
| 4 | Most of the algorithm are known | Technology still being developed for certain format |
| 5 | Cryptography alter the structure of the secret message | Steganography does not alter the structure of the secret message |

**Table 1 Cryptography Vs Steganography**

**1.1 Motivation**

Information security main intension is to provide the security to our information. there are some methods where user can make his information secure they are cryptography and Steganography.

* Cryptography generally change the plain text in to cipher text.
* Intruder tries to break the cipher with some algorithm.
* It may take several days but at last it can be cracked which should not be done.
* Steganography is the method of hiding the information. The intruder don’t know the existence.
* Steganography is widely used so many of them including the intuder know the methodology.
* So, there are several methods were implemented in steganography among text-to-text based steganography is new.
* So, the intruder have very less chance to identify this.

**1.2 Problem Definition**

The information should be secure which we want to convey so this tool helps in doing that with simple limited words.

**1.3 Objective Of Project**

The main objective was to create a tool so that the information will be hidden securely and the person who is on other side with the tool can decode it.

**1.4 Organization of Project**

* Steganography tool is used in order to make the file information secure.
* In this we have used ASCII values and UK words so the input message will be mapped.
* In the receiver side this process will be in reverse i.e, UK word to ASCII value.

**CHAPTER – 2**

**LITERATURE SURVEY**

**2.1 Introduction**

**2.2 Existing Systems**

**2.3 Disadvantages of Existing System**

**2.4 Proposed System**

**CHAPTER-3**

**REQUIREMENT SPECIFICATION**

To run this project on various platforms we need some hardware and software to support this project.

**3.1 Software Requirements**

* Eclipse IDE or Bluej
* JDK(latest version)
* JRE(latest version)

**3.2 Hardware Requirements**

* Processor: Pentium IV.
* Speed: 2.7 GHZ.
* Primary Memory: 256 MB RAM.
* Hard Disk: 25GB

**CHAPTER-4**

**MODULES DESCRIPTION**

There are two modules in Text Steganography tool. They are:

* Sender
* Receiver

**4.1.SENDER:**

Sender is the one who want to input the original message file. when he inputs the file then encoding method is string a parameter will be invoked and further substitutions are made based on message and further the cover-text.txt file will be generated.

**4.2. RECEIVER:**

Receiver is the one to whom we want to share our cover-text.txt so, receiver will input the cover-file.txt then ASCII characters will be substituted instead of uk words and further generates the original-text.txt.

These are the two modules present in the project so we have developed these things in very easy and clearly understandable way. So, by seeing the above description the proposed system having many advantages as compared to the existing system.

**CHAPTER 5**

**DESIGN AND ANALYSIS**

**5.1 UML DIAGRAMS**

## What is a UML Diagram?

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development.

## What is Meant by UML?

UML stands for Unified Modeling Language. UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

* Improved integration between structural models like class diagrams and behavior models like activity diagrams.
* Added the ability to define a hierarchy and decompose a software system into components and subcomponents.
* The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed statechart diagrams to state machine diagrams, also known as state diagrams.

## Types of UML Diagrams

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing, and deployment.

These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

*Structural UML diagrams*

* Class diagram
* Package diagram
* Object diagram
* Component diagram
* Composite structure diagram
* Deployment diagram

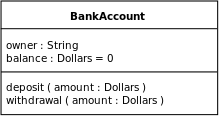
*Behavioral UML diagrams*

* Activity diagram
* Sequence diagram
* Use case diagram
* State diagram
* Communication diagram
* Interaction overview diagram

**Class Diagram**

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering), a **class diagram** in the [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of static structure diagram that describes the structure of a system by showing the system's [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) modeling. It is used for general [conceptual modeling](https://en.wikipedia.org/wiki/Conceptual_model) of the systematic of the application, and for detailed modeling translating the models into [programming code](https://en.wikipedia.org/wiki/Programming_code). Class diagrams can also be used for [data modeling](https://en.wikipedia.org/wiki/Data_modeling).[[1]](https://en.wikipedia.org/wiki/Class_diagram#cite_note-1) The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.



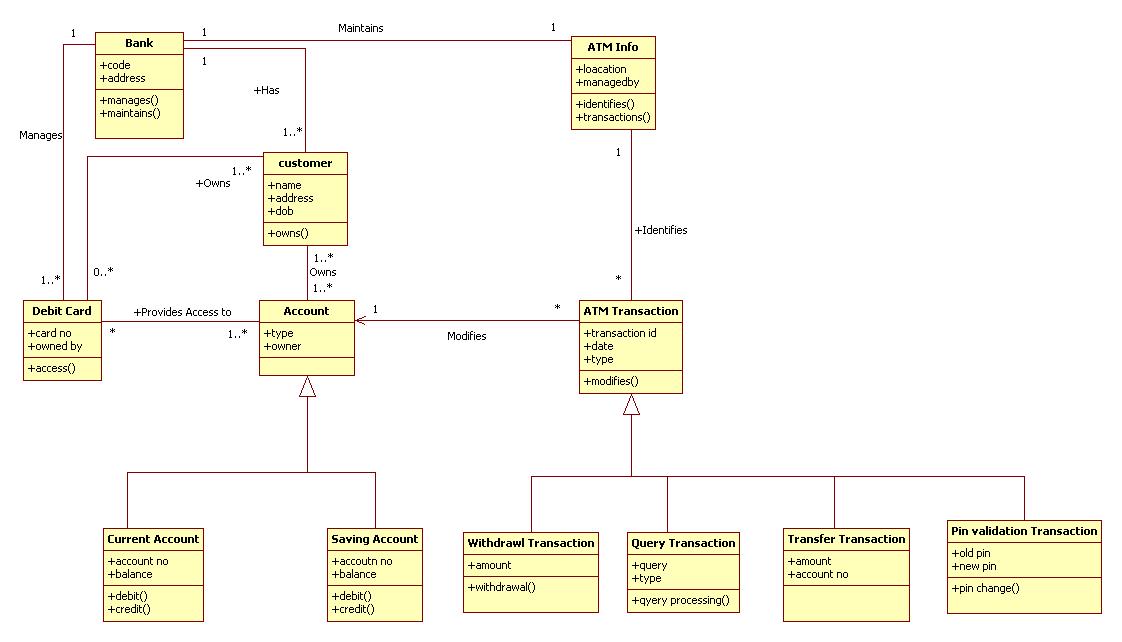
A class with three compartments.

In the diagram, classes are represented with boxes that contain three compartments:

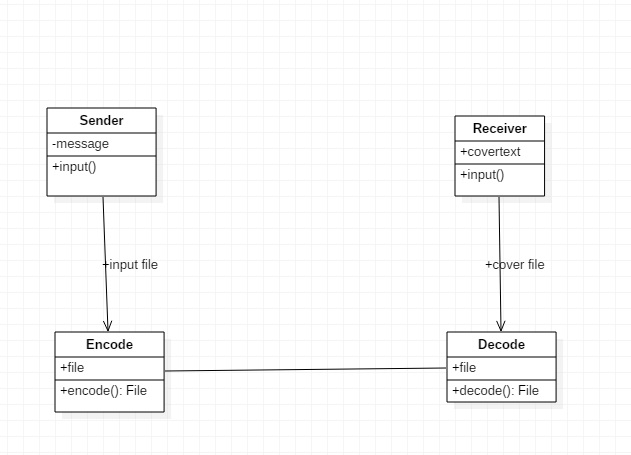
* The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
* The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
* The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

In order to further describe the behavior of systems, these class diagrams can be complemented by a [state diagram](https://en.wikipedia.org/wiki/State_diagram) or [UML state machine](https://en.wikipedia.org/wiki/UML_state_machine).[[2]](https://en.wikipedia.org/wiki/Class_diagram#cite_note-SWA09-2)

****

Example of ATM machine Class diagram



**Figure 5.1.1: CLASS DIAGRAM: Text Steganography tool**

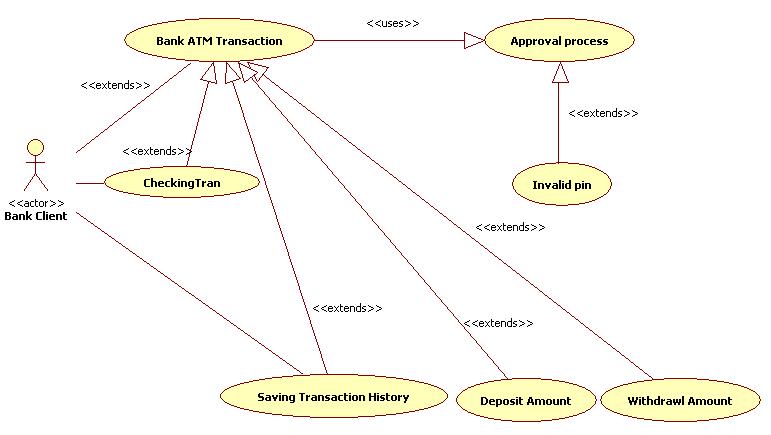
**USECASE DIAGRAM**

Use case diagram consists of use cases and actors and shows the interaction between the use cases and actors. Use cases are the function that are to be performed in the module. An actor could be the end user of the system or external system.**Use case diagrams** are usually referred to as [**behavior diagrams**](http://www.uml-diagrams.org/uml-25-diagrams.html#behavior-diagram) used to describe a set of actions ([**use cases**](http://www.uml-diagrams.org/use-case.html)) that some system or systems ([**subject**](http://www.uml-diagrams.org/use-case-subject.html)) should or can perform in collaboration with one or more **external users** of the system ([**actors**](http://www.uml-diagrams.org/use-case-actor.html)). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

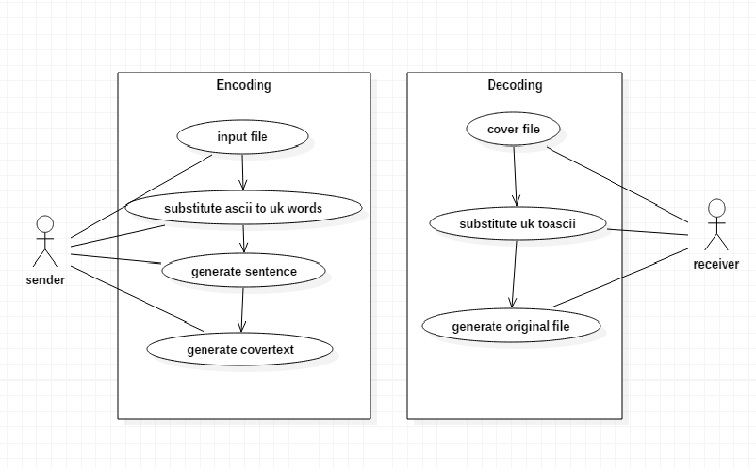
Note, that UML 2.0 to 2.4 specifications also described **use case diagram** as a specialization of a [**class diagram**](http://www.uml-diagrams.org/class-diagrams-overview.html), and class diagram is a [**structure diagram**](http://www.uml-diagrams.org/uml-25-diagrams.html#structure-diagram).

Use case diagrams are in fact twofold - they are both [**behavior diagrams**](http://www.uml-diagrams.org/uml-25-diagrams.html#behavior-diagram), because they describe behavior of the system, and they are also [**structure diagrams**](http://www.uml-diagrams.org/uml-25-diagrams.html#structure-diagram) - as a special case of class diagrams where classifiers are restricted to be either [**actors**](http://www.uml-diagrams.org/use-case-actor.html) or [**use cases**](http://www.uml-diagrams.org/use-case.html) related to each other with [**associations**](http://www.uml-diagrams.org/association.html).

[[UML 2.5 FTF - Beta 1]](http://www.uml-diagrams.org/references.html#ref-uml-25-b1) moved use cases out of behavior modeling to UML **supplementary concepts**. So, it is an unfortunate quandary what kind of UML diagrams use case diagrams are.



Example of ATM machine Usecase diagram



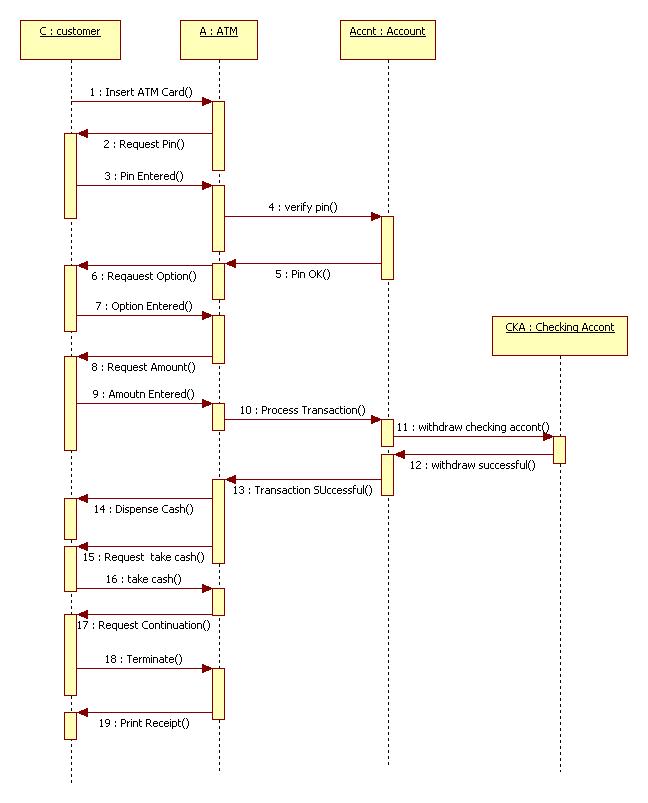
**Figure 5.1.2: USECASE DIAGRAM: Text Steganography tool**

**SEQUENCE DIAGRAM**

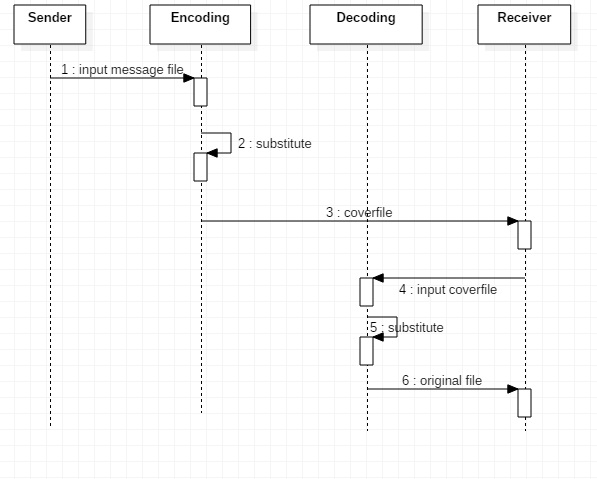
UML sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom, the interaction starts near the top of the diagram and ends at the bottom. A popular use for them is to document the dynamics in an object oriented system. It is a construct of a [message sequence chart](https://en.wikipedia.org/wiki/Message_sequence_chart).

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



Example of ATM machine Sequence diagram



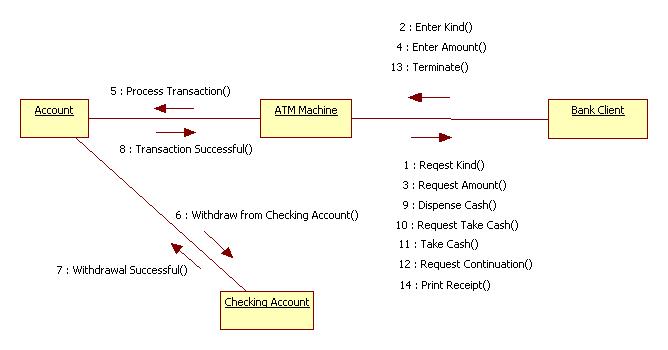
**Figure 5.1.3: SEQUENCE DIAGRAM: Text Steganography tool**

**COLLABORATION DIAGRAM**

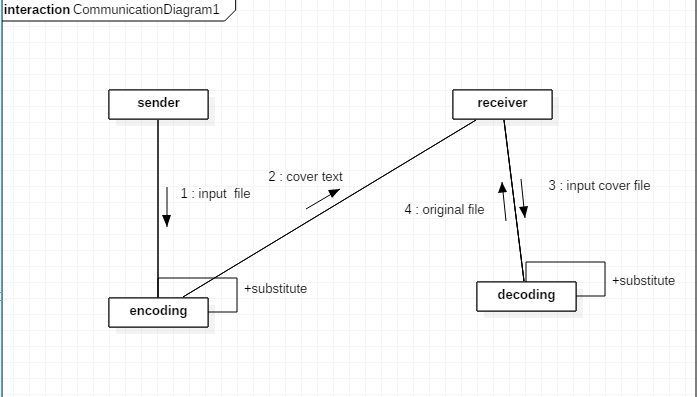
A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult to read. Several vendors offer software for creating and editing collaboration diagrams.



Example of ATM machine Collaboration diagram

**Figure 5.1.4: COLLABORATION DIAGRAM: Text Steganography tool**

**ACTIVITY DIAGRAM**

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

## Purpose of Activity Diagrams:

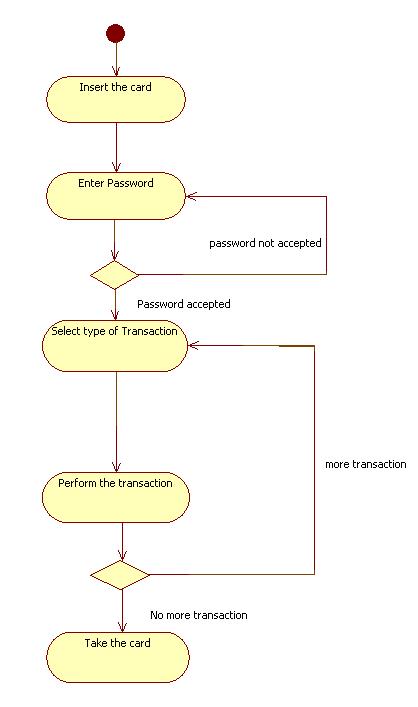
The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

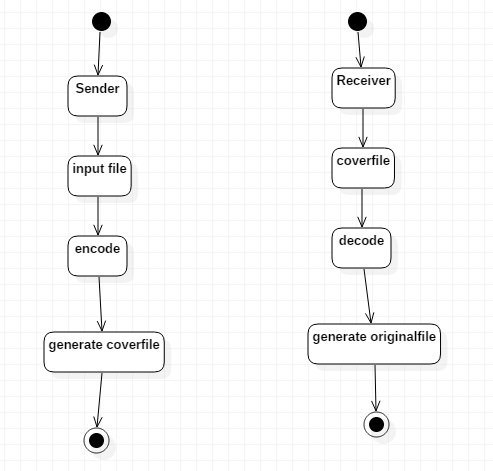
It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as −

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.



Example of ATM machine Activity diagram

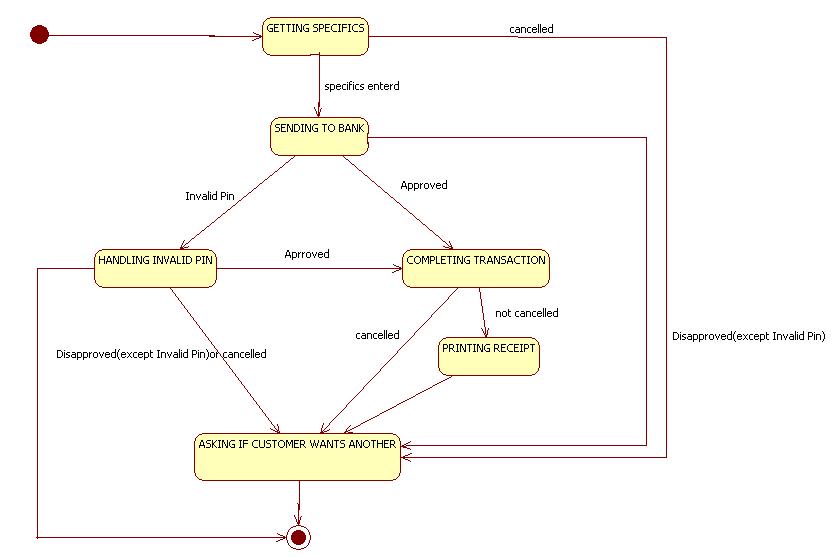
****

**Figure 5.1.5:ACTIVITY DIAGRAM: Text Steganography tool**

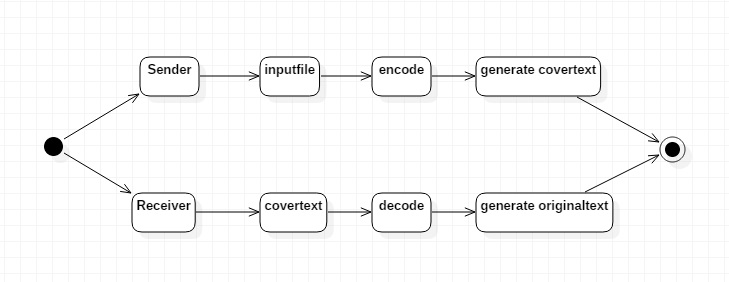
**STATECHART DIAGRAM**

A state diagram, also called a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity. State diagrams are useful in all forms of object-oriented programming (OOP). The concept is more than a decade old but has been refined as OOP modeling paradigms have evolved.

A state diagram resembles a [flowchart](http://whatis.techtarget.com/definition/flowchart) in which the initial state is represented by a large black dot and subsequent states are portrayed as boxes with rounded corners. There may be one or two horizontal lines through a box, dividing it into stacked sections. In that case, the upper section contains the name of the state, the middle section (if any) contains the state [variable](http://whatis.techtarget.com/definition/variable)s and the lower section contains the actions performed in that state. If there are no horizontal lines through a box, only the name of the state is written inside it. External straight lines, each with an arrow at one end, connect various pairs of boxes. These lines define the transitions between states. The final state is portrayed as a large black dot with a circle around it. Historical states are denoted as circles with the letter H inside.



Example of ATM machine State diagram

****

**Figure 5.1.6: STATECHART DIAGRAM: Text Steganography tool**

**COMPONENT DIAGRAM**

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executable, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

## Purpose of Component Diagrams:

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

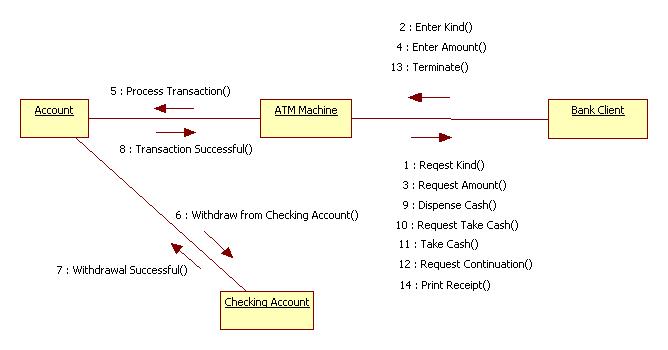
Thus from that point of view, component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc.

Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

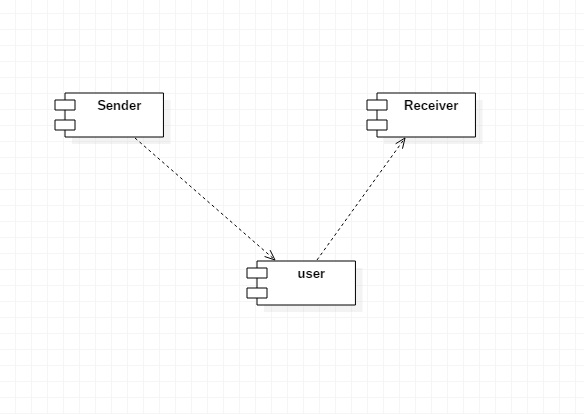
A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as −

* Visualize the components of a system.
* Construct executable by using forward and reverse engineering.
* Describe the organization and relationships of the components.

****

Example of ATM machine Component diagram

****

**Figure 5.1.7: COMPONENT DIAGRAM: Text Steganography tool**

**DEPLOYMENT DIAGRAM**

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

## Purpose of Deployment Diagrams

The term Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components, where software components are deployed. Component diagrams and deployment diagrams are closely related.

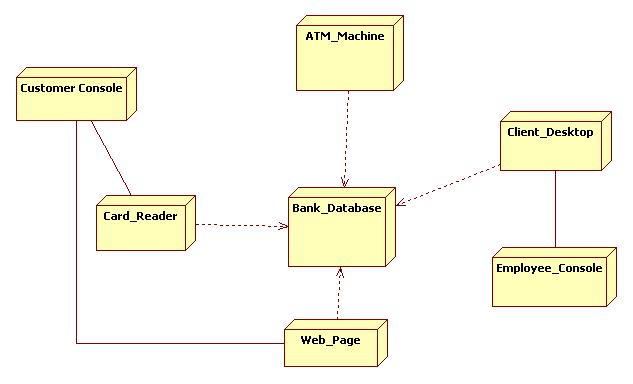
Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.

UML is mainly designed to focus on the software artifacts of a system. However, these two diagrams are special diagrams used to focus on software and hardware components.

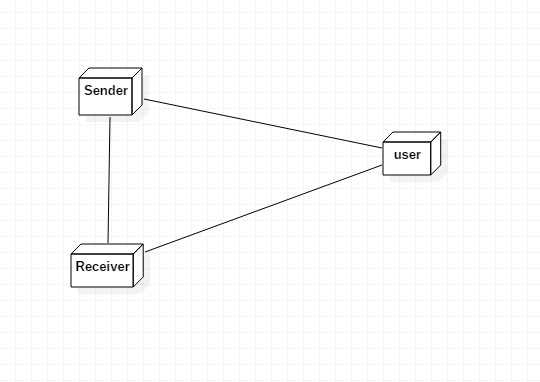
Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system. Deployment diagrams are used by the system engineers.

The purpose of deployment diagrams can be described as −

* Visualize the hardware topology of a system.
* Describe the hardware components used to deploy software components.
* Describe the runtime processing nodes.

****

Example of ATM machine Deployment diagram

****

**Figure 5.1.8: DEPLOYMENT DIAGRAM: Text Steganography tool**

**CHAPTER 6**

**IMPLEMENTATION AND RESULTS**

**6.1 INTRODUCTION**

The Steganography is the hiding of information in an of media. in that the Text Steganography is based on hiding text inside a text. Text Steganography contains some methods to implement in that substitution is one the method and we have implemented our algorithm based on substitution method. In general we are taking complete ascii list and uk words of ascii length and we are substituting the ascii value to the uk word and further we are making a sentence to every word so that it is more tough to identify.

**6.2 SOURCE CODE:**

**stego.java**

/\*\*

\* @author bhargav

\* @version 1.0 19-09-2017

\*/

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.\*;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.util.Random;

import java.io.\*;

import java.lang.\*;

import java.lang.StringBuilder;

import java.lang.StringBuffer;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JFileChooser;

//import java.util.Base64;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.filechooser.FileSystemView;

import javax.xml.bind.DatatypeConverter;

/\* Name of the class has to be "Main" only if the class is public. \*/

public class stego extends JFrame implements ActionListener {

private static final String NULL = null;

final static int NO\_WORDS = 5;

final static int NO\_SENTS = 20;

final static String SPACE = " ";

final static String PERIOD = ".";

File file=null;

File file1=null;

static Random r = new Random();

// String inpstr="bhargav";

private static String str = "";

private static String strtemp = "";

final JFileChooser fc1 = new JFileChooser("F:/java projects/steganography");

final JFileChooser fc2 = new JFileChooser("F:/java projects/steganography");

JTextField t2;

JButton sender, receiver, jb1, jb2;

JLabel lb,heading;

public stego() {

JFrame fr = new JFrame("Steganography");

Image icon = Toolkit.getDefaultToolkit().getImage("icon.png");

fr.setIconImage(icon);

try {

fr.setContentPane(new JLabel(new ImageIcon(ImageIO.read(new File("background.jpg")))));

} catch (IOException n) {

n.printStackTrace();

}

/\*

\* lb=new JLabel("enter text to hide"); lb.setBounds(50,20,100,30);

\* t2=new JTextField();

\*

\*

\* t2.setBounds(50,50,200,30); b=new JButton("Click Here");

\* b.setBounds(50,100,95,30);

\*

\* //lb1.setBounds(50,100,200,30); fr.add(t2); fr.add(b); fr.add(lb);

\* //fr.add(lb1); fr.setSize(400,400);

\* fr.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

\* fr.setLayout(null); fr.setVisible(true); b.addActionListener(new

\* ActionListener() {

\*

\* @Override public void actionPerformed(ActionEvent arg0) { // TODO

\* Auto-generated method stub try { str=t2.getText();

\* steganography\_and\_encryption(); JOptionPane.showMessageDialog(null,

\* "Sucessfully completed\nFiles Generated Are:\n\tcovertext.txt\ncipherfile\ndecipherfile"

\* ); // //System.out.println(str); } catch(Exception e) { }

\*

\*

\* } });

\*/

heading = new JLabel("Text Steganography tool");

heading.setBounds(130, 20, 150, 30);

heading.setFont(new Font("Serif", Font.BOLD, 14));

heading.setForeground(Color.RED);

sender = new JButton("Sender");

sender.setBounds(150, 100, 95, 30);

sender.setBackground(Color.GREEN);

receiver = new JButton("Receiver");

receiver.setBounds(150, 200, 95, 30);

receiver.setBackground(Color.YELLOW);

fr.add(heading);

fr.add(sender);

fr.add(receiver);

fr.setSize(400, 400);

fr.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

fr.setLayout(null);

fr.setVisible(true);

sender.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

JFrame panel = new JFrame("Encode");

Image icon = Toolkit.getDefaultToolkit().getImage("icon.png");

panel.setIconImage(icon);

try {

panel.setContentPane(new JLabel(new ImageIcon(ImageIO.read(new File("background.jpg")))));

} catch (IOException n) {

n.printStackTrace();

}

JLabel lb = new JLabel("input file !!!!!!");

lb.setBounds(50, 20, 100, 30);

lb.setFont(new Font("Serif", Font.BOLD, 14));

lb.setForeground(Color.RED);

final JTextField tf = new JTextField(20);

tf.setBounds(50, 50, 300, 20);

jb1 = new JButton("RESET");

jb1.setBounds(50, 80, 100, 25);

jb1.setBackground(Color.YELLOW);

jb2 = new JButton("Browse");

jb2.setBounds(200, 80, 100, 25);

jb2.setBackground(Color.GREEN);

// tf.setB

panel.add(lb);

panel.add(tf);

panel.add(jb1);

panel.add(jb2);

panel.setSize(400, 400);

panel.setLayout(null);

panel.setVisible(true);

jb1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

tf.setText("");

}

});

jb2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

//System.out.println("hghghghg");

try

{

int returnVal = fc1.showOpenDialog(stego.this);

if (returnVal == JFileChooser.APPROVE\_OPTION) {

file = fc1.getSelectedFile();

//This is where a real application would open the file.

tf.setText(file.getName());

}

BufferedReader in = new BufferedReader(new FileReader(file));

// if(in.readLine() == null){

// JOptionPane.showMessageDialog(null, "File is empty!!! choose another");

/// }

// else {

String line = in.readLine();

if(line != null){

while(line != null){

str += line;

line = in.readLine();

}

//str=file.toString();

System.out.println(str);

steganography\_and\_encryption(str);

JOptionPane.showMessageDialog(null, "Sucessfully completed\nFiles Generated Are:\n\tcovertext.txt\ncipherfile.txt\nconfirm.txt");

}

else

JOptionPane.showMessageDialog(null, "File is empty!!! choose another");

//

//System.out.println(str);

}

catch(Exception ex)

{

}

/\* JOptionPane.showMessageDialog(null,

"This language just gets better and better!");\*/

}

});

}

});

receiver.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg1) {

JFrame panel1 = new JFrame("Decode");

Image icon = Toolkit.getDefaultToolkit().getImage("icon.png");

panel1.setIconImage(icon);

try {

panel1.setContentPane(new JLabel(new ImageIcon(ImageIO.read(new File("background.jpg")))));

} catch (IOException n) {

n.printStackTrace();

}

JLabel lb1 = new JLabel("Cover Text....!!!");

lb1.setBounds(50, 20, 100, 30);

lb1.setFont(new Font("Serif", Font.BOLD, 14));

lb1.setForeground(Color.RED);

final JTextField tf1 = new JTextField(20);

tf1.setBounds(50, 50, 300, 20);

JButton jb3 = new JButton("RESET");

jb3.setBounds(50, 80, 100, 25);

jb3.setBackground(Color.YELLOW);

JButton jb4 = new JButton("Browse");

jb4.setBounds(200, 80, 100, 25);

jb4.setBackground(Color.GREEN);

// tf.setB

panel1.add(tf1);

panel1.add(jb3);

panel1.add(jb4);

panel1.setSize(400, 400);

panel1.add(lb1);

panel1.setSize(400, 400);

panel1.setLayout(null);

panel1.setVisible(true);

//strtemp=str;

jb3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

tf1.setText("");

}

});

jb4.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

try

{

//still to do

int returnVal1 = fc2.showOpenDialog(stego.this);

if (returnVal1 == JFileChooser.APPROVE\_OPTION) {

file1 = fc2.getSelectedFile();

//This is where a real application would open the file.

tf1.setText(file1.getName());

}

BufferedReader in = new BufferedReader(new FileReader(file));

String line = in.readLine();

if(line != null){

while(line != null){

strtemp += line;

line = in.readLine();

}

// strtemp=strtemp.replace("", " ").trim();

// strtemp=file1.toString();

// steganography\_and\_encryption(str);

//System.out.println(str);

// JOptionPane.showMessageDialog(null, "Sucessfully completed\nFiles Generated Are:\n\tcovertext.txt\ncipherfile\ndecipherfile");

System.out.println("decoding starts here");

decrypt\_and\_decode(strtemp);

JOptionPane.showMessageDialog(null, "Message is in Originalfile!");

}

else{

JOptionPane.showMessageDialog(null, "File is empty!!! choose another");

}

//JOptionPane.showMessageDialog(null,"This language just gets better and better!");

}

catch(Exception ex)

{

}

}

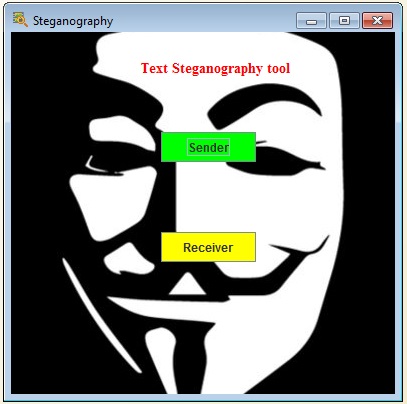
});

}

});

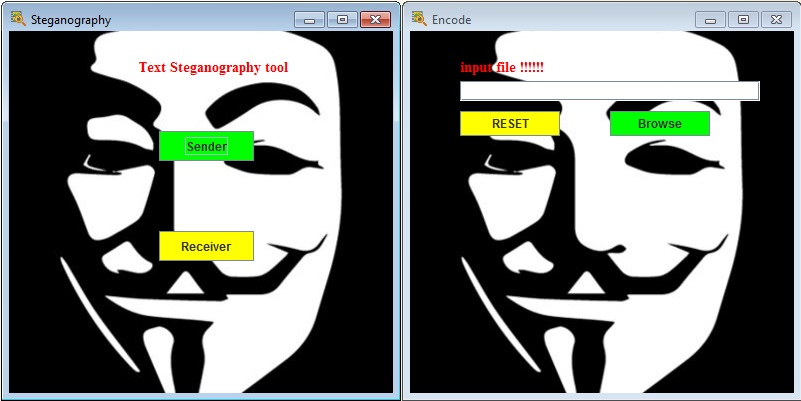
}

**6.3 SCREENSHOTS**

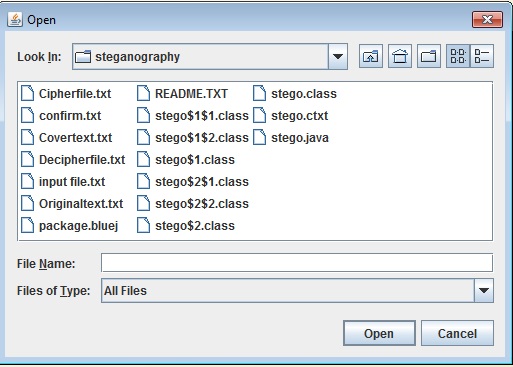
****

Screen 6.3.1: Main Frame

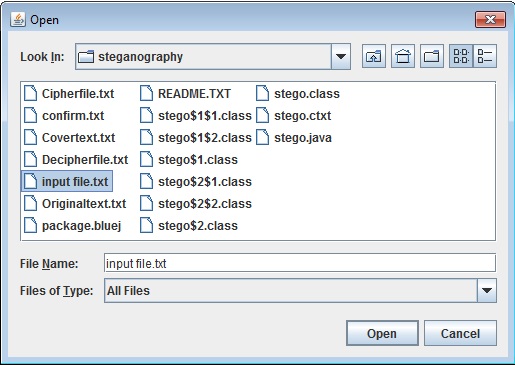
**Sender**

****

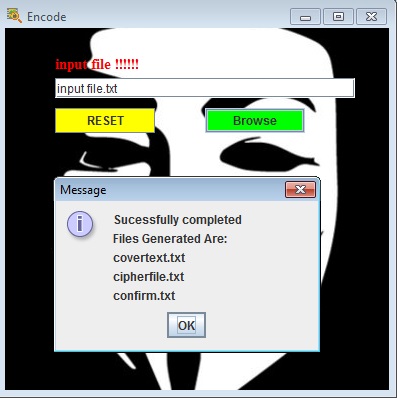
Screen 6.3.2: sender frame

****

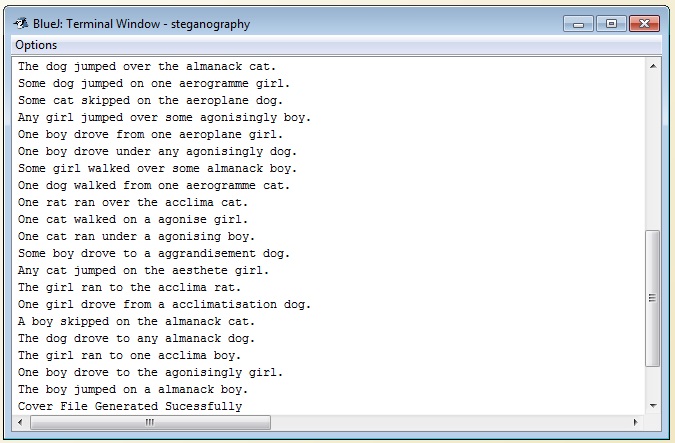
Screen 6.3.3: sender file chooser

****

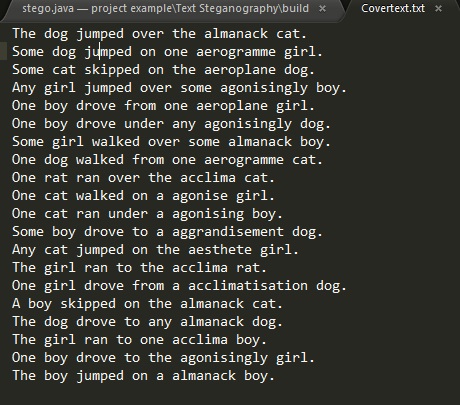
Screen 6.3.4: sender input file



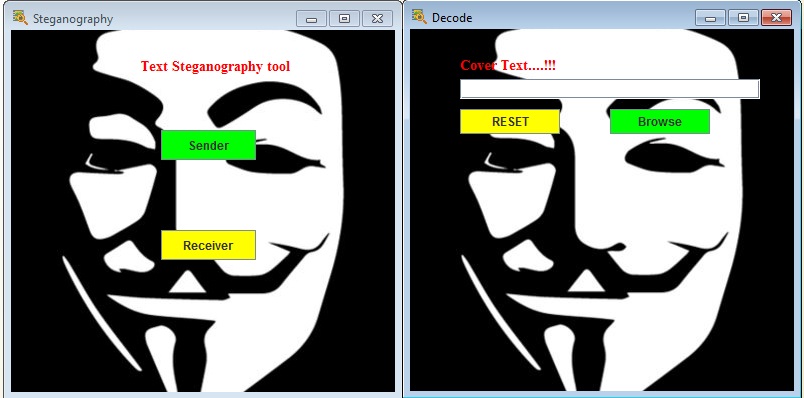
Screen 6.3.5: sender success

****

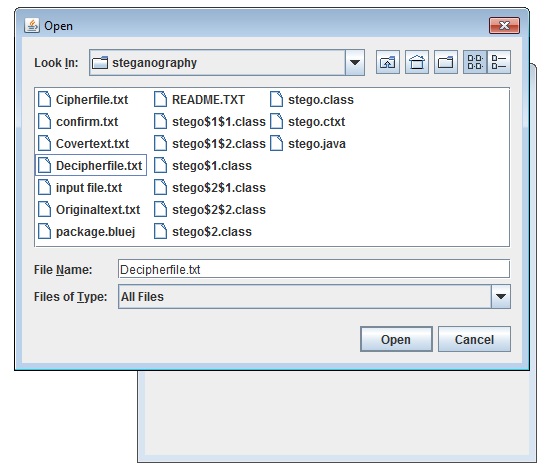
Screen 6.3.6: sender terminal

****

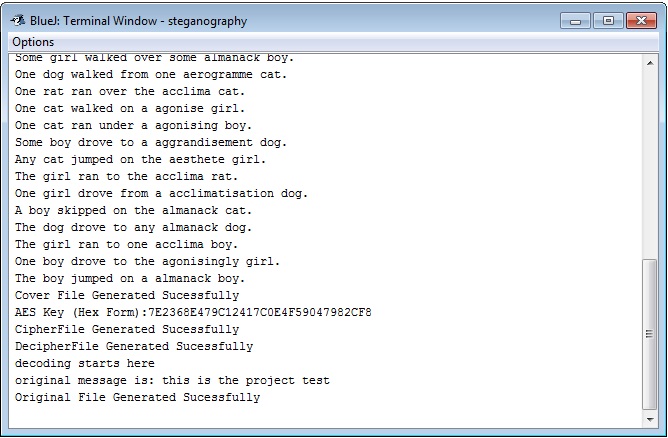
Screen 6.3.7: coverfile

****

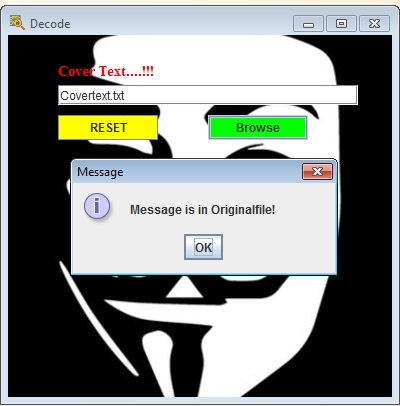
Screen 6.3.8: Receiver Frame

****

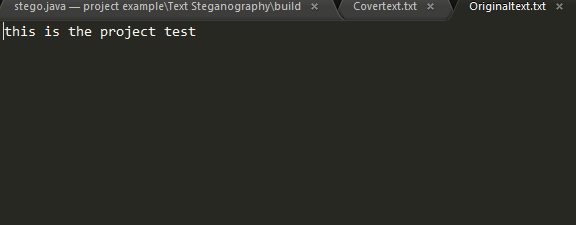
Screen 6.3.9: receiver filechooser

****

Screen 6.3.10: receiver terminal

****

Screen 6.3.11: receiver success

****

Screen 6.3.12: view result

**CHAPTER - 7**

**TESTING AND VALIDATION**

**7.1 INTRODUCTION**

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

**TESTING OBJECTIVES**

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has probability of finding an error, if it exists.
* The test is inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

**Levels of Testing:** In order to uncover present in different phases we have the concept of levels of testing.

**7.2 DESIGN OF TEST CASES AND SCENARIOS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test**  **Case No.** | Input | Expected Behavior | **Observed**  **behavior** | **Status**  **P = Passed**  **F = Failed** |
| 1 | Choose the  input file | Input file must be selected | -do- | P |
| 2 | Input file should  have message | Must show the warning message and prevent to run | -do- | P |
| 3 | Apply Encoding logic | Must Successfully generate cover file, confirm file, cipher file | -do- | P |
| 4 | Cover file | Cover file must contain some sentences | -do- | P |
| 5 | Input cover file for Decoding | Cover file must be inputted | -do- | P |
| 6 | Check cover file | Must show the warning message and prevent to run | -do- | P |
| 7 | Apply Decoding logic | Must successfully generate Original file | -do- | P |

**THE BASIC LEVELS OF TESTING**

Client needs acceptance testing

Requirements system testing 

Design integration testing

Code unit testing

**Specification Testing:**

Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested.

**Unit testing:**

In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system.

Each Module can be tested using the following two Strategies:

1. Black Box Testing

**BLACK BOX TESTING**

**What is Black Box Testing?**

Black box testing is a software testing techniques in which **functionality of the software under test (SUT) is tested without looking at the internal code structure**, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.**In Black Box Testing we just focus on inputs and output of the software system** without bothering about internal knowledge of the software program.           

The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

**Black box testing - Steps**

Here are the generic steps followed to carry out any type of Black Box Testing.

* Initially requirements and specifications of the system are examined.
* Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
* Tester determines expected outputs for all those inputs.
* Software tester constructs test cases with the selected inputs.
* The test cases are executed.
* Software tester compares the actual outputs with the expected outputs.
* Defects if any are fixed and re-tested.

**Types of Black Box Testing**

There are many types of Black Box Testing but following are the prominent ones -

* **Functional testing** – This black box testing type is related to functional requirements of a system; it is done by software testers.
* **Non-functional testing** – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements  such as performance, scalability, usability.
* **Regression testing** – Regression testing is done  after code fixes , upgrades or any other system maintenance to check the new code has not affected the existing code.

**CHAPTER-8**

**8.1 PROJECT CONCLUSION**

This project is design in order to provide the security to the file containing text. In The wide range of network when we are communicating and wanted to share some of the privacy or important information. We cannot specify that the message is only read by specific one so, there may be any third party who was watching all our messages that contain privacy information so this tool helps in increase the information security by hiding the original text. so, the third party may think this is not much important information and ignores. but the specific one whom we want to share can able to decode it with the tool.

**8.2 FUTURE ENHANCEMENT**

This cannot be specified as complete because in the information security if the flaw is identified by attackers they just don’t leave until they try to crack the hidden or encrypted text so more and more security need to be provided. in this project the words we have considered are uk vs us words so that we can get a limited number so if the attacker know that so he may try to brute force with every occurrence. so, this is just a basic implementation and can further be extended.