

CHAPTER 1

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.

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.

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:

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

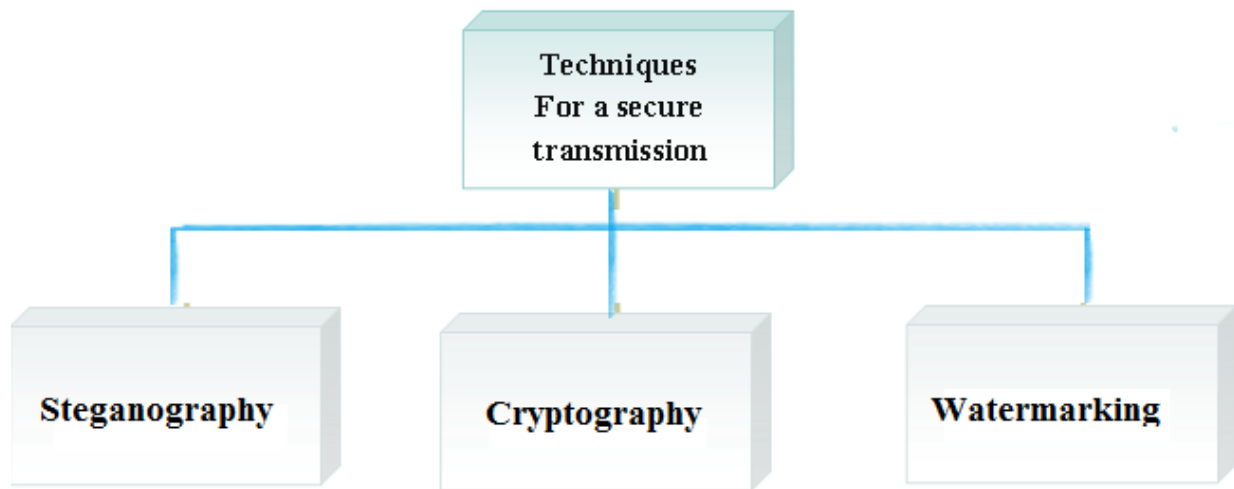


Fig 1: Secure transmission techniques

1. Cryptography
2. Steganography
3. Watermarking

1.1 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. 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:

- 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 of the original plain text from this hash value.

1.2 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.

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).

1.3 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.

1.4 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.

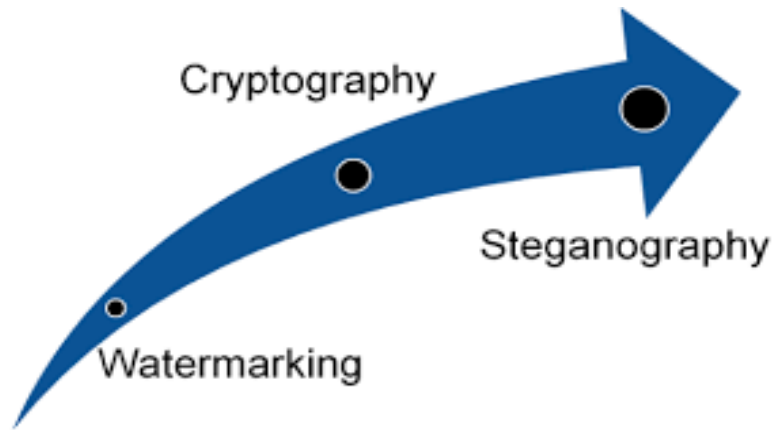


Fig 2: Watermarking to Cryptography to Steganography

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

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

1.2 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 changes 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 doesn't know the existence.
- Steganography is widely used so many of them including the intruder know the methodology.
- So, there are several methods were implemented in Steganography among text-to-text based Steganography is new.
- So, the intruders have very less chance to identify this.

1.3 Problem Definition

The information should be secure which we want to convey. So, in this project we have developed an algorithm that maps with limited UK words that differs from US words.

1.4 Objective of Project

The main objective was to make the information secure. This can be achieved with the implemented algorithm.

1.5 Organization of Project

This includes 8 chapters:

- Chapter 1:
Motivation of about what made us think of the problem and find the solution to the problem has been shown in this section of the documentation. It also includes the brief description of the Problem Statement.
- Chapter 2
The Existing System and its disadvantages have been explained, leading to the Proposed System giving a glance of what the project is about.
- Chapter 3

The project is a nothing without a prototype. A list of what all is the requirements for the project have been given in this section along with a brief description of each of them.

- Chapter 4

For the ease of the project, it's being divided into modules, each module having its own priority and operations to be performed. These modules are being explained along with their actions.

- Chapter 5

Every project is first described and shown in the form of a diagram where the whole process of the actions in the project is explained in prior. Here the Process and the Data flow are being described and shown in the form of Diagrams.

- Chapter 6

Explains about how the things described in the above sections are implemented using the prescribed languages and tools and the overall result of the project is being told.

- Chapter 7

Explains about how the validation of the project and how the testing is being performed and what are the test cases followed are described.

- Chapter 8

Concludes about the overall project and future enhancements are also discussed in this chapter.

CHAPTER 2

LITREATURE SURVEY

2.1 INTRODUCTION

Steganography is a Greek word which means concealed writing. The word “steganos” means “covered “ and “graphical “ means “writing” . Thus, steganography is not only the art of hiding data but also hiding the fact of transmission of secret data. Steganography hides the secret data in another file in such a way that only the recipient knows the existence of message.

In ancient time, the data was protected by hiding it on the back of wax, writing tables, and stomach of rabbits or on the scalp of the slaves. But today’s most of the people transmit the data in the form of text, images, video, and audio over the medium. In order to safely transmission of confidential data, the multimedia object like audio, video, images are used as a cover sources to hide the data.

A general Steganography system is shown in Figure 3. It is assumed that the sender wishes to send via Steganographic transmission, a message to a receiver. The sender starts with a cover message, which is an input to the stegosystem, in which the embedded message will be hidden. The hidden message is called the embedded message. A Steganographic algorithm combines the cover message with the embedded message, which is something to be hidden in the cover (Nspw, 2006).The algorithm may, or may not, use a Steganographic key (stego key), which is additional secret data that may be needed in the hidden process.

The same key (or related one) is usually needed to extract the embedded message again. The output of the Steganographic algorithm is the stego message. The cover message and stego message must be of the same data type, but the embedded message may be of another data type. The receiver reverses the embedding process to extract the embedded message (Avedissian, 2005). fig3

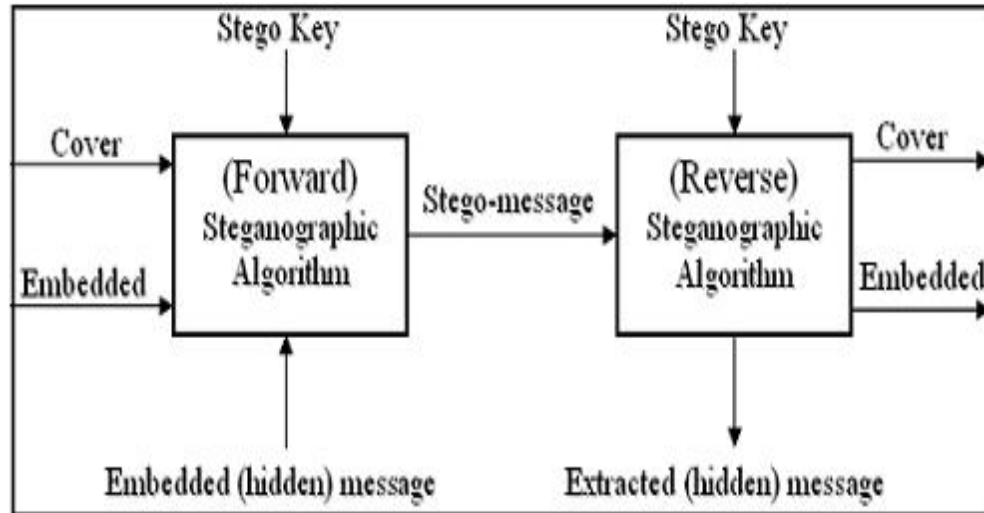


Fig 3: Steganography Working Model

2.1.1 Types of Steganography

The following fig 4 shows the different type's Steganography approaches

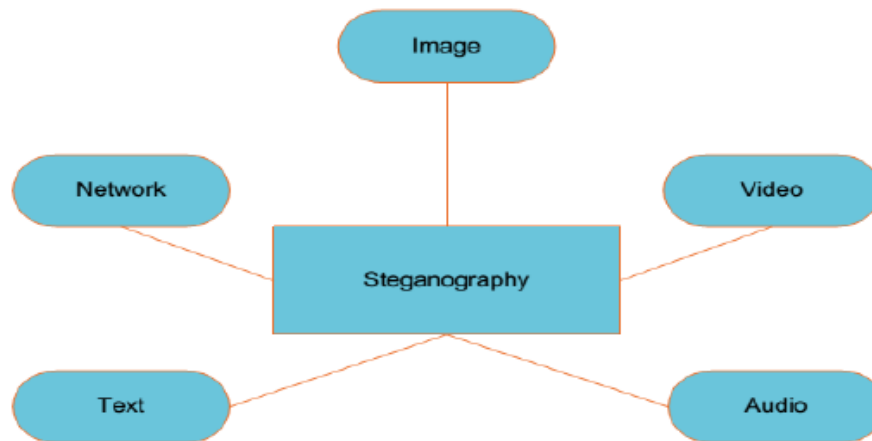


Fig 4: Types of Steganography

2.1.1.1 Text Steganography: It consists of hiding information inside the text files. In this method, the secret data is hidden behind every nth letter of every words of text message. Numbers of methods are available for hiding data in text file. These methods are

- i) Format Based Method;
- ii) Random and Statistical Method;
- iii) Linguistics Method.

2.1.1.2 Image Steganography: Hiding the data by taking the cover object as image is referred as image steganography. In image steganography pixel intensities are used to hide the data. In digital steganography, images are widely used cover source because there are number of bits presents in digital representation of an image.

2.1.1.3 Audio Steganography: It involves hiding data in audio files. This method hides the data in WAV, AU and MP3 sound files. There are different methods of audio steganography. These methods are

- i) Low Bit Encoding
- ii) Phase Coding
- iii) Spread Spectrum.

2.1.1.4 Video Steganography: It is a technique of hiding any kind of files or data into digital video format. In this case video (combination of pictures) is used as carrier for hiding the data. Generally discrete cosine transform (DCT) alter the values (e.g., 8.667 to 9) which is used to hide the data in each of the images in the video, which is unnoticeable by the human eye. H.264, Mp4, MPEG, AVI are the formats used by video steganography.

2.1.1.5 Network or Protocol Steganography: It involves hiding the information by taking the network protocol such as TCP, UDP, ICMP, IP etc, as cover object. . In the OSI layer network model there exist covert channels where steganography can be used.

The present work is focused on text steganography and is discussed in the following sections

2.1.1.1 TEXT STEGANOGRAPHY

Text steganography involves anything like changing the format of an existing text, changing words within a text, generating random character sequences. Due to deficiency of redundant information which is present in image, audio or a video file, text steganography is believed to be the trickiest technique. In text documents, we can hide information by introducing changes in the structure of the document without making a notable change in the concerned output.

Unperceivable changes can be made to an image or an audio file, but, in text files, even an additional letter or punctuation can be marked by a casual reader. Storing text file require less memory and its faster as well as easier communication makes it preferable to other types of Steganographic methods. Text steganography can be broadly classified into three types as shown in the fig 5: Format based Random and Statistical generation, Linguistic methods.

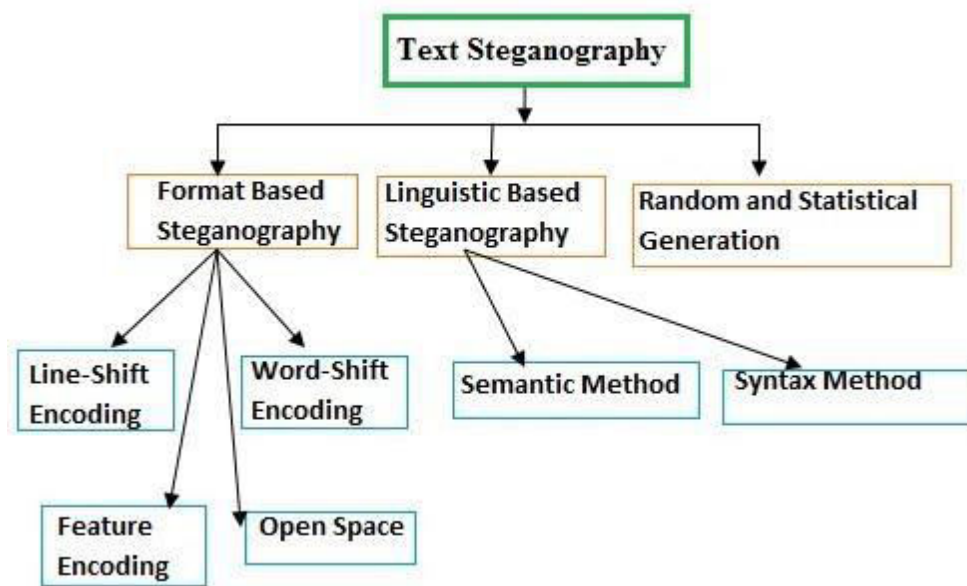


Fig 5: Types of Text Steganography

2.1.1.1.1 Format-based method

Format-based methods usually modify existing text for hiding the steganographic text. Insertion of spaces or non displayed characters, careful errors tinny throughout the text and resizing of fonts are some of the many format based methods used in text steganography. There are many of the techniques been in text steganography, for implementation of thesis word mapping technique is introduced to provide two way secure data which encrypts a secret message using genetic operator crossover and then embeds the resulting cipher text, taking two bits at a time, in a cover file by inserting blank spaces between words of even or odd length using a certain mapping technique. The embedding positions are saved in another file and transmitted to the receiver along with the stego object.

2.1.1.1.2 Random and Statistical generation

This avoid comparison with a known plaintext, steganographers often resort to generating their own cover texts. Character sequences method hide the information within character sequences.

2.1.1.1.3 Linguistic method

Linguistic method is a combination of syntax and semantics methods. Linguistic Steganography considers the linguistic properties of generated and modified text, and uses linguistic structure as the space in which messages are hidden.

Linguistic Steganography Types

- 1 Semantic Method
- 2 Syntactical Method

2.1.1.1.4 Semantic Method

This method is implemented by introducing a change in the meaning of the text. Semantic method takes into account the synonyms of a word. The synonyms convey the same meaning so they can be used in a better way to hide a message. For example: primary and secondary meanings of a word can be used in a text to hide a message. This will prevent attacker from knowing that he is reading a cover text. Semantic method is the one which does not destroys the hidden information even if an Optical Character Recognition technique is used. Even OCR is not able to detect the hidden message

2.1.1.1.5 Syntactic Method:

Syntactical method as the name suggests focuses on the syntax of the text. The syntax of the text can be varied by inserting punctuations marks or by using different spellings of a word. As an example, comma (,) or full stop(.) can be used to hide secret information. Another example can be that American and British English change the spellings of the words. It is a better technique to use these languages for creating a stego object. The chance of

detection of hidden message by the attacker is minimal. The attraction of the attacker will not be attracted if syntactical methods are used in an appropriate manner.

2.1.2 Types of Text Steganography Techniques

1. Line shift
2. Word shift
3. White Steg
4. Spam text
5. Syntactic method
6. Word mapping
7. CSS (Cascading Style Sheets)
8. Mixed-case font
9. SMS Texting
10. Feature Coding
11. Cricket match Scorecard
12. Missing letter puzzle
13. Hiding data in wordlist
14. Hiding data in paragraphs
15. Hiding data in HTML
16. Hiding data in letter points and extensions.

2.1.3 Text Steganography Tools

Within Text Steganography area there are many tools as shown in the table 2.

Text Steganography Tools	Plain Text	Other	Source Code	License	Production
NiceTtext	Yes		Yes	Open Source	Yes
Snow	Yes		Yes	Open Source	Yes
Textto	Yes		Yes	Opensource	Yes
Sam's Big Play marker	Yes		Yes	Open source	Yes
Steganosaurus	Yes		Yes	Opensource	Yes
Steganous	Yes	HTML		Commecial	Yes

Mimic	Yes			Opensource	Yes
-------	-----	--	--	------------	-----

Table 2 Steganography Tools

2.2 LITERATURE REVIEW

Many research papers have been reviewed and some of them have been discussed below

2.2.1 Review Paper 1: Text Steganography in SMS

Authors: Mohammad Shirali-Shahreza, M.Hassan Shirali-Shahreza

2.2.1.1 Problem Taken:

A new method for hiding secret information, which is written by using daily used abbreviation in sending SMS for example ASAP(As soon as possible) is used.

2.2.1.2 Methodology/Approach Applied:

Authors suggested for substituting words with their abbreviations or vice versa to hide bits of secret message during CHAT. The SMS-Texting language is a combination of abbreviated words used in SMS. These words can be also used in abbreviation text Steganography method. For this purpose, the words and phrases that have abbreviated forms are identified in the SMS. These words may be ordinary words, such as University which has the known abbreviation of Univ. or may be a word from the collection of words of SMS-Texting, for example, the word "you" with its abbreviated form of "u". As described above, by using full or abbreviated form of words or phrases, the information are hidden in the text. Extraction of information is done by reverse operations. In this method, not only using SMS words attracts no attention but also one has an increased choice of words, because in addition to ordinary abbreviations, the abbreviated phrases common in SMS are also used.

The proposed method is composed of two phases:

- 1- Steganography phase which has the duty of hiding information in the SMS
- 2- Extractor phases which has the duty of extracting information from the SMS.

A list of abbreviated words and phrases of SMS Texting were prepared from the “SMS Testing”. This list had fields with two values: one full word and the other its abbreviated

form. If a word had more than one abbreviation, the abbreviations were separated by comma (.). These two lists are then merged and used as a single list for two phases, the Steganography and the extractor phases. In preparing list of SMS-Texting words, words coined by some people for special purposes can also be used.

The Steganography phase searches the SMS for the words existing in the list according to the algorithm described. If the number of words found were less than the length of array of zero and one bits which made from the data we want to hide, the phase will not hide the data in the given SMS and that the size of information given is big. Otherwise, it acts according to the algorithm and uses the full word in the text for hiding bit 0 and the abbreviated form for hiding bit 1 as shown table 3. Thus, the information is hidden in the SMS. At the end, the SMS is sent to the consignee whose number is received from the user and on a special port.

After receiving the SMS on the recipient's mobile phone, the extractor phase identifies the words by the use of the list of words with abbreviated forms and stores the zero or one value in an array based on the full or abbreviated form of the words. The hidden information is extracted.

Acronym	Translation	Explanation
2l8	Too late	The time is too late, missed opportunity
ASAP	As Soon As Possible	Immediately
C	See	Do you understand? OR the verb 'to see'
CM	Call Me	Asking someone to telephone
F2F	Face to face	In person
NC	No Comment	I can't say what I think
R	are	The verb 'to be'
SRY	Sorry	An apology
T+	Think positive	You need to be positive about a situation
ZZZZ	Sleeping	I'm tired, bored or annoyed

Table 3: SMS Texting

2.2.1.3 Resolved Issues/Conclusion:

Authors proposed a new method for Steganography in SMS message using abbreviation text Steganography method, which can be used other devices such as pocket PC PDAs etc, which can also be implemented on desktop PCs using SMS gateway for sending and receiving

SMS messages. Millions of SMS are being exchanged daily which makes the task of steg-SMS analysis difficult.

2.2.1.4 Unresolved Issues/Future Scope:

The approach is confined to SMS texting language which can be varied for different persons. Official communications like letters from companies cannot use this approach.

The approach can be used for sending small amount of data only as SMS itself is an intention of sending Short messages. The capacity of the technology is very low, and the embedding concentrated only on the abbreviations of the text.

2.2.2 Review Paper 2: Adaptation of Text Steganographic Algorithms for HTML

Authors: Stanislav S.Barilink, igor V.Minin, Oleg V.Minin

2.2.2.1 Problem Taken:

The number of users of the internet has increased which is open to everyone where information hiding can be done through HTML tags

2.2.2.2 Methodology/Approach Applied:

Use of special steganographic technologies which enable placing within web-document (HTML- codes) "invisible" for unauthorized person messages will help to solve this problem at least partly. Bits of hidden information are introduced in a form of unprintable symbols. Such symbols are a "gap" and a "horizontal tabulation". Authors have represented bits in the form of symbols: "1" - gap, "O" - "horizontal tabulation". Each byte of hidden information is transformed into a succession of these symbols where each symbol corresponds with a bit of hidden byte.

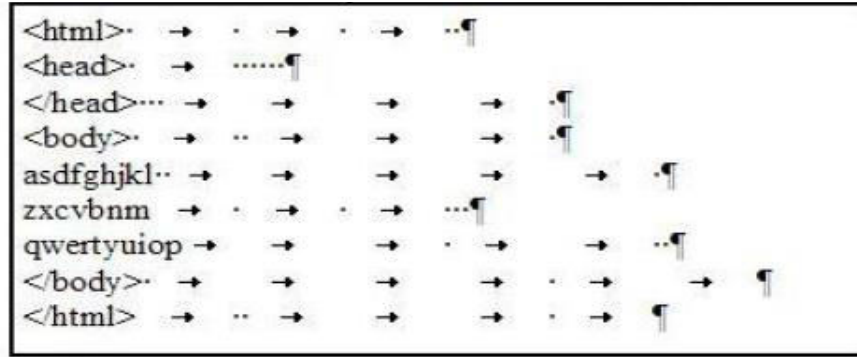


Fig 6: HTML file

2.2.2.3 Resolved Issues/Conclusion:

Authors proposed a new approach of hiding data in html files

2.2.2.4 Unresolved Issues/Future Scope:

Text-editor in the mode of representation of unprintable symbols (T - switching this mode in Microsoft word). << >> is a "gap", «->» is "horizontal tabulation", « T >> is a line-shift. Thus, editor shows "visibility" of hidden information.

2.2.3 Review Paper 3: Text Steganography Based On Font Type in Ms-Word Documents

Authors: Wesam Bhaya, Abdul Monem Rahma And Dhamyaa Al-Nasrawi

2.2.3.1 Problem Taken:

A new method of a novel text Steganography method which takes into account the Font Types is considered, this new method depends on the Similarity of English Font Types called as (SEFT) technique. It works by replace font by more similar fonts. The secret message was encoded and embedded as similar fonts in capital Letters of cover document.

2.2.3.2 Approach:

Authors proposed "Similar English Font Type" (SEFT). In this method for writing hidden messages in text of document file format (which lack of redundancy compared to images or audio) called (Similar English Font Types, SEFT, Technique) use the most similarity types of English fonts in hiding message by changing the font to another. In general, any type of font has

many of types similar to its front types. This property is the basic of their study as show in fig 8 and 9.

Even if the font is changed to different form for example Arial changed to Arial Unicode MS or Geo_Arial or Microsoft San Serf which looks similar to a normal human eye, for example

Hell World→Arial

Hell World→ Arial Unicode MS

Hell World→ Microsoft San Serf

It essentially consists of four main components:

1. Create similar font array:

This is the most important component of the method. Begin by determines the type of document font and then find the more similar types of it. In this study, assumed (15) type of cover document fonts which are more usable and prevalent in text documents (TXT, MS Word, PDF, PPT).

2. Create code Table:

In this Authors prescribed a font as in table format and designed code to identity the fonts for the given text and theirs similar fonts. The table format and code table as follows

Index	Font name		Similarity	
1	Arial	Arial Unicode MS	Geo_Arial	Microsoft Sans Serif
2	Book Antiqua	Palatino Linotype	Antiqua	Caudex
3	Candara	Khmer UI	Ebrima	Microsoft New Tai Lue
4	Century	Century751 BT	CenturyOldStyle	CenturyExpd BT
5	Calibri	Gisha	Leelawadee	Liberation Serif
6	Cambria	Proforma	EideticNeoRegular	Liberation Serif
7	Comic Sans	SF Toontime	Komika Text	SF Arch Rival Extended
8	Times New Roman	Tinos	Liberation Serif	Thorndale
9	Helvetica	Arimo	Geo_Arial	Arial-Relcom
10	Courier New	Courier New CE	TiredOfCourier	TiredOfCourierThin
11	Verdana	Tahoma	MS Reference Sans Serif	Lato
12	Perpetua	ChanticleerRoman	Centaur	CaslonOldFace BT
13	Lucida Sans	Lucida Sans Unicode	Segoe UI	Lucida Sans Typewriter
14	Thorndale	Times New Roman	Liberation Serif	Tinos
15	Franklin Gothic Book	Ebrima	Corbel	Trebuchet MS

Fig 7: Font Types in MS Word

Index	Characters	F1	F2	F3	index	Characters	F1	F2	F3
1	a	1	1	1	16	P	2	3	1
2	b	1	1	2	17	Q	2	3	2
3	c	1	1	3	18	R	2	3	3
4	d	1	2	1	19	S	3	1	1
5	e	1	2	2	20	T	3	1	2
6	f	1	2	3	21	u	3	1	3
7	g	1	3	1	22	v	3	2	1
8	h	1	3	2	23	w	3	2	2
9	I	1	3	3	24	x	3	2	3
10	j	2	1	1	25	y	3	3	1
11	k	2	1	2	26	z	3	3	2
12	l	2	1	3	27	space	3	3	3
13	m	2	2	1					
14	n	2	2	2					
15	o	2	2	3					

Fig 8: Codes for the above font table

3. Embedding process:
 - a. Open cover document, find its type of font
 - b. Scan cover document to find capitals English letters,
 - c. Compute number of capitals English letters to check the capability of embedding
 - d. For each symbol in secret message
 - e. Retrieve its code
 - f. Change font type of three capitals
4. Extracting process
 - a. Open Stego document
 - b. For each three capitals letters
 - c. Determine the code
 - d. If the code is (0, 0, 0), then the end of secret
 - e. message was reached
 - f. Else, find corresponding secret symbol, using code table

2.2.3.3 Resolved Issues/Conclusion:

Authors proposed a novel method of hiding information in Microsoft Word documents. Microsoft Word documents are very much common in everyday life of today's digital world. The capacity of this method is very high, depending on the number of Capital Letters in cover document.

2.2.3.4 Unresolved Issues/Future Scope:

This approach is not a robust approach; if MS word does not support the respective font format type then the message is lost which may be altered due to change in the versions of the MS word document. This approach is confined to only MS word.

2.3 PROBLEM DEFINATION

By considering the above three existing system we can identify certain flaws.

Some of them were

- Paper 3: The approach can be used for sending small amount of data only as SMS itself is an intention of sending Short messages. The capacity of the technology is very low, and the embedding concentrated only on the abbreviations of the text.
- Paper 2: Complex in usage and understanding of the algorithm.
- Paper 3: Microsoft Word documents are very much common in everyday life of today's digital world. The capacity of this method is very high, depending on the number of Capital Letters in cover document.

2.4 PROPOSED SYSTEM

By considering the literature research and taking in to consideration of limitations, we proposed an algorithm "Text Steganography By Changing Word Spellings Tool". In the following project we have considered the above limitations and developed an algorithm.

- The complexity and understanding of proposed algorithm is easy because limited words i.e., differently spelt UK and US words are taken.
- The proposed algorithm is applicable for large amount text (file) rather than short text.
- The proposed algorithm is applicable complete ASCII values so, every inputted text can be hidden.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

3.1 SOFTWARE REQUIREMENT

A software requirement is a field within software engineering that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as:

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.
4. The activities related to working with software requirements can broadly be broken down into elicitation, analysis, specification, and management.

3.1.1 Software requirements

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

3.2 HARDWARE REQUIREMENT

Computer hardware refers to the physical parts of a computer and related devices. Internal hardware devices include motherboards, hard drives, and RAM. External hardware devices include monitors, keyboards, mice, printers, and scanners.

The internal hardware parts of a computer are often referred to as components, while external hardware devices are usually called peripherals. Together, they all fall under the category of computer hardware. Software, on the other hand, consists of the programs and applications that run on computers. Because software runs on computer hardware, software programs often have system requirements that list the minimum hardware required for the software to run.

3.2.1 Hardware requirements

Hardware and software requirements may vary depending on the machine and operating system.

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

CHAPTER 4

MODULE DESCRIPTION

4.1 INTRODUCTION

In this project the Sender and Receiver are important modules because the Sender is the one who want to make his information secure. Similarly the Receiver from the other end wants to decode the hidden text which was shared by sender.

There are two modules in Text this project. They are:

- Sender
- Receiver

4.2 SENDER:

Sender is the one who want to make this information secure. When user inputs the file then encoding method invoked and further substitutions are made based on message and further the cover-text.txt file will be generated.

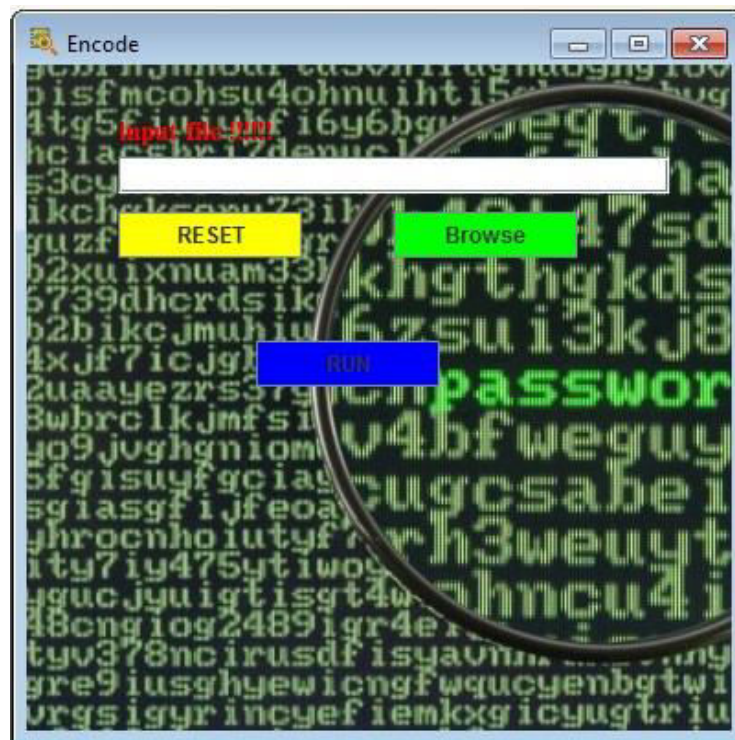


Fig 9: Sender

4.3 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. Considering the above description the proposed system having many advantages as compared to the existing system.



Fig 10: Receiver

CHAPTER 5

DESIGN AND ANALYSIS

5.1 INTRODUCTION

In the present scenario, information security is major issue we don't know if there is any third party who are viewing our sensitive information. So, in order to achieve it text steganography is best approach among others and having very less chances to identify. And with minimal limitations this algorithm can overcome some of the major flaws.

5.2 SYSTEM ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A description of the design and contents of a computer system. If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software. A formal description of a system, or a detailed plan of the system at component level to guide its implementation. Architecture comprises the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.

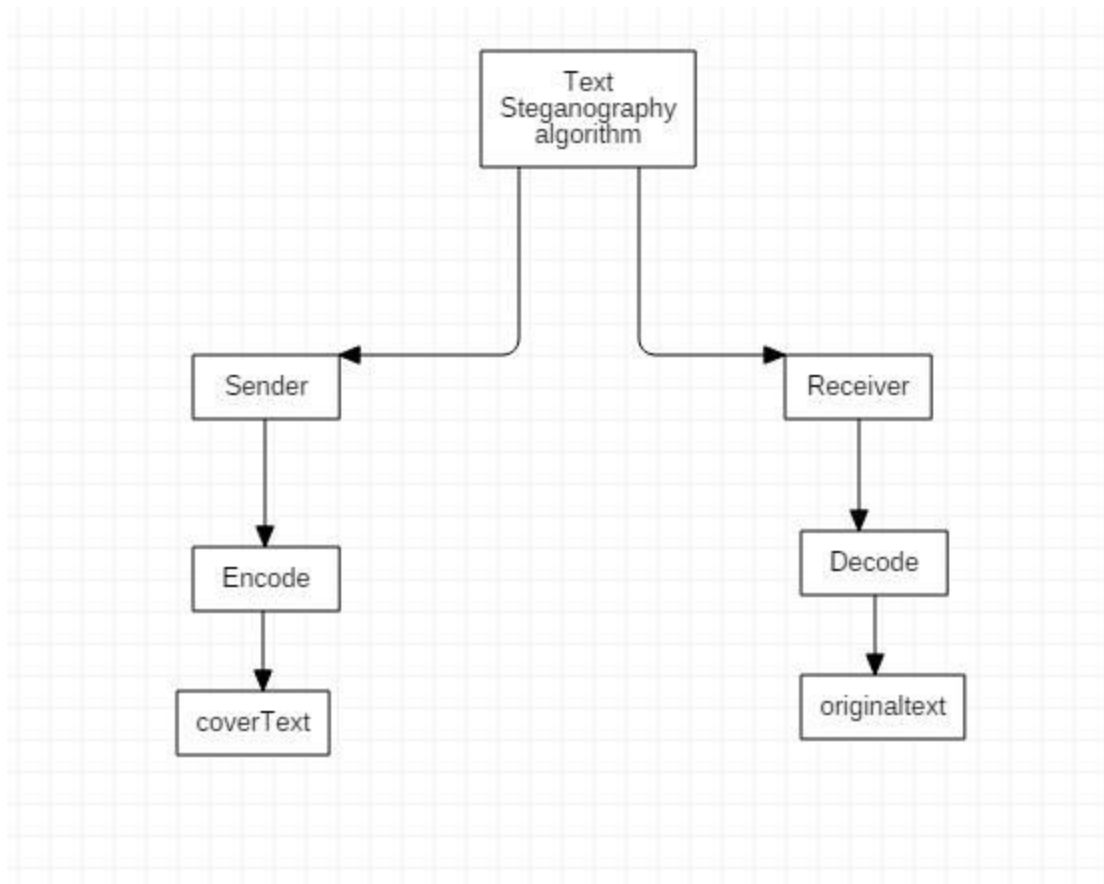


Fig 11: System Architecture

5.3 ENTITY RELATIONSHIP DIAGRAM

An entity-relationship diagram (ERD) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An ERD is a conceptual and representational model of data used to represent the entity framework infrastructure.

The elements of an ERD are:

- Entities
- Relationships
- Attributes

Steps involved in creating an ERD include:

1. Identifying and defining the entities
2. Determining all interactions between the entities
3. Analyzing the nature of interactions/determining the cardinality of the relationships
4. Creating the ERD
5. An entity-relationship diagram (ERD) is crucial to creating a good database design. It is used as a high-level logical data model, which is useful in developing a conceptual design for databases.
6. An entity is a real-world item or concept that exists on its own. Entities are equivalent to database tables in a relational database, with each row of the table representing an instance of that entity.
7. An attribute of an entity is a particular property that describes the entity. A relationship is the association that describes the interaction between entities. Cardinality, in the context of ERD, is the number of instances of one entity that can, or must, be associated with each instance of another entity. In general, there may be one-to-one, one-to-many, or many-to-many relationships.
8. For example, let us consider two real-world entities, an employee and his department. An employee has attributes such as an employee number, name, department number, etc. Similarly, department number and name can be defined as attributes of a department. A department can interact with many employees, but an employee can belong to only one department, hence there can be a one-to-many relationship, defined between department and employee.
9. In the actual database, the employee table will have department number as a foreign key, referencing from department table, to enforce the relationship.

5.3.1 E-R Diagram for <Text Steganography tool> :

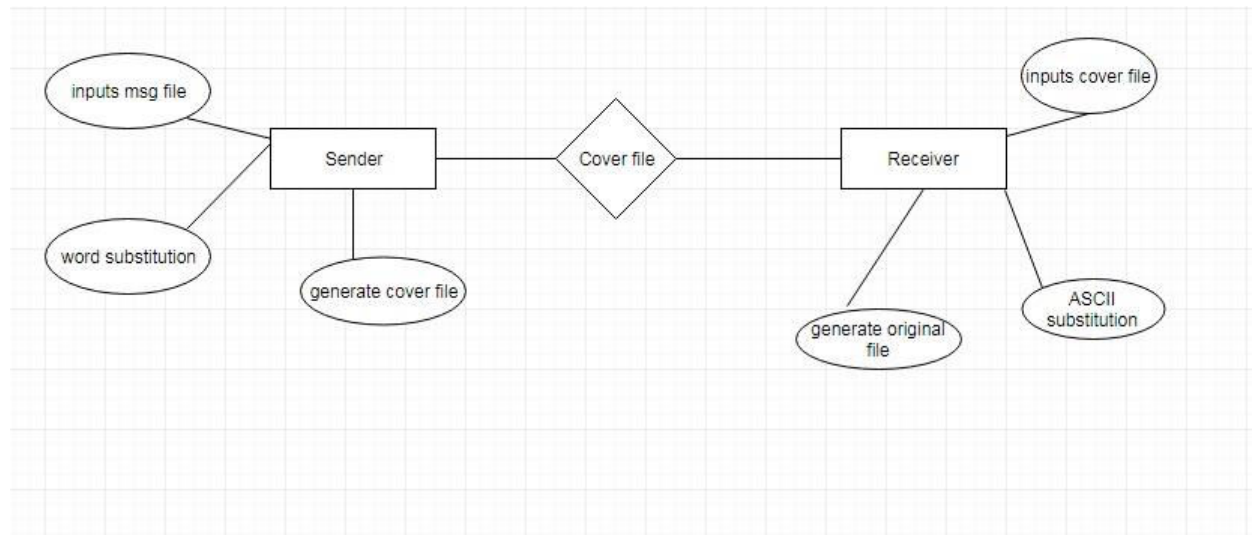


Fig 12: E-R Diagram

5.4 UML DIAGRAMS

5.4.1 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.

5.4.2 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.

5.4.3 Building blocks of UML:

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

5.4.4 Things in the UML:

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things
- An-notational things

5.4.4.1 Structural things are the nouns of UML models. The structural things used in the project design are:

First, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

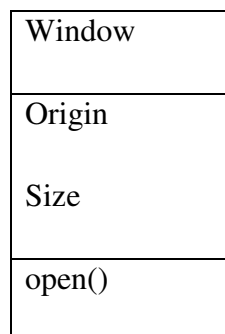


Fig 13: Structural things

Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.

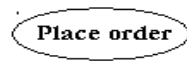


Fig 14: use case structure

Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.

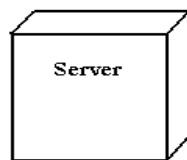


Fig 15: use case structure

5.4.4.2 Behavioral things are the dynamic parts of UML models. The behavioral thing used is:

5.4.4.2.1 Interaction:

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).

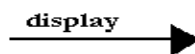


Fig 16: Interaction

5.4.5 Relationships in the UML:

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

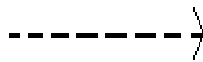


Fig 17: Dependency

An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.



Fig 18: Association

A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).



Fig 19: Generalization

A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

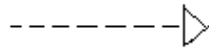


Fig 20: Realization

5.4.6 Sequence Diagrams:

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

5.4.6.1 Actor

Represents an external person or entity that interacts with the system

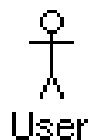


Fig 21 Actor

5.4.6.2 Object

Represents an object in the system or one of its components

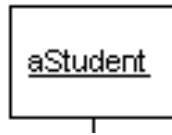


Fig 22 Object

5.4.6.3 Unit

Represents a subsystem, component, unit, or other logical entity in the system (may or may not be implemented by objects)

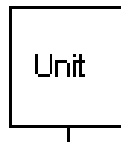


Fig 23 unit

5.4.6.4 Separator

Represents an interface or boundary between subsystems, components or units (e.g., air interface, Internet, network)



Fig 24 Separator

5.4.6.5 Group

Groups related header elements into subsystems or components

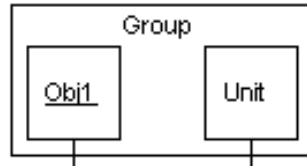


Fig 25 Group

5.4.7 Sequence Diagram Body Elements

5.4.7.1 Action

Represents an action taken by an actor, object or unit



Fig 26 Action

5.4.7.2 Asynchronous Message

An asynchronous message between header elements

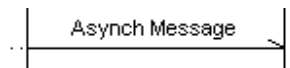


Fig 27 Asynchronous Message

5.4.7.3 Block

A block representing a loop or conditional for a particular header element

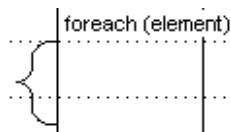


Fig 28 Block

5.4.7.4 Call Message

A call (procedure) message between header elements

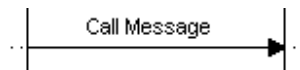


Fig 29 Call Message

5.4.7.5 Create Message

A "create" message that creates a header element (represented by lifeline going from dashed to solid pattern)

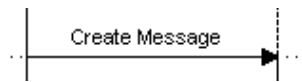


Fig 30 Create Message

5.4.7.6 Diagram Link

Represents a portion of a diagram being treated as a functional block. Similar to a procedure or function call that abstracts functionality or details not shown at this level. Can optionally be linked to another diagram for elaboration.

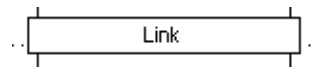


Fig 31 Diagram Link

Else Block Represents an "else" block portion of a diagram block

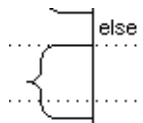


Fig 32 Diagram Link

5.4.7.7 Message

A simple message between header elements



Fig 33 Message

5.4.7.8 Return Message

A return message between header elements

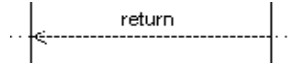


Fig 34 Return Message

To understand the UML, you need to form a conceptual model of the language, and this requires learning three major elements: the UML's basic building blocks, the rules that dictate how those building blocks may be put together, and some common mechanisms that apply throughout the UML. Once you have grasped these ideas, you will be able to read UML models and create some basic ones. As you gain more experience in applying the UML, you can build on this conceptual model, using more advanced features of the language.

5.4.8 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

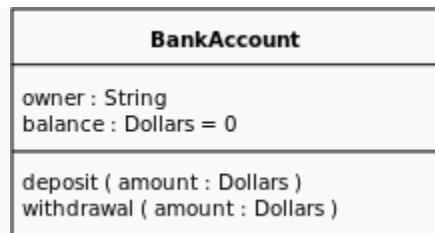
- Activity diagram
- Sequence diagram
- Use case diagram

- State diagram
- Communication diagram
- Interaction overview diagram

5.4.8.1 Class Diagram

In software engineering, a **class diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. 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 or UML state machine.

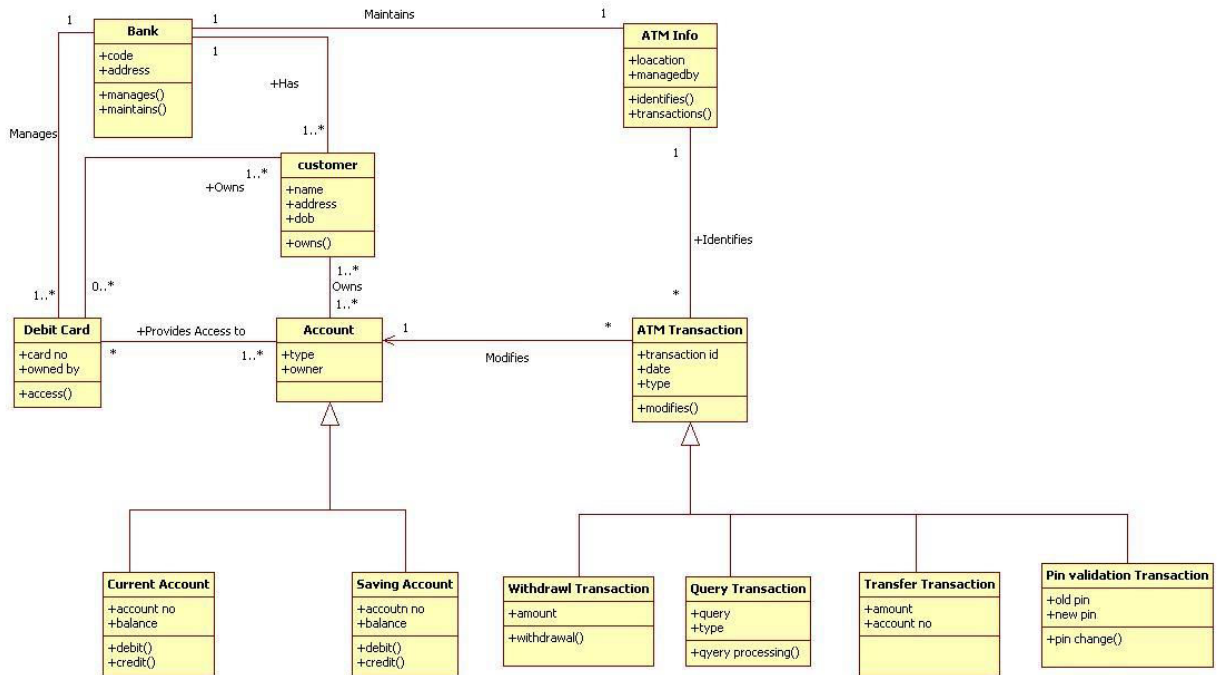


Fig 35 Atm class diagram

5.4.8.1.1 CLASS DIAGRAM FOR PROJECT

Consider the above Fig-35 as reference the class diagram consist of two main class Sender and the Receiver and the Encode and Decode consist of two main methods those are encode() and decode() in which the file is taken as input from the sender side and further the covertext file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

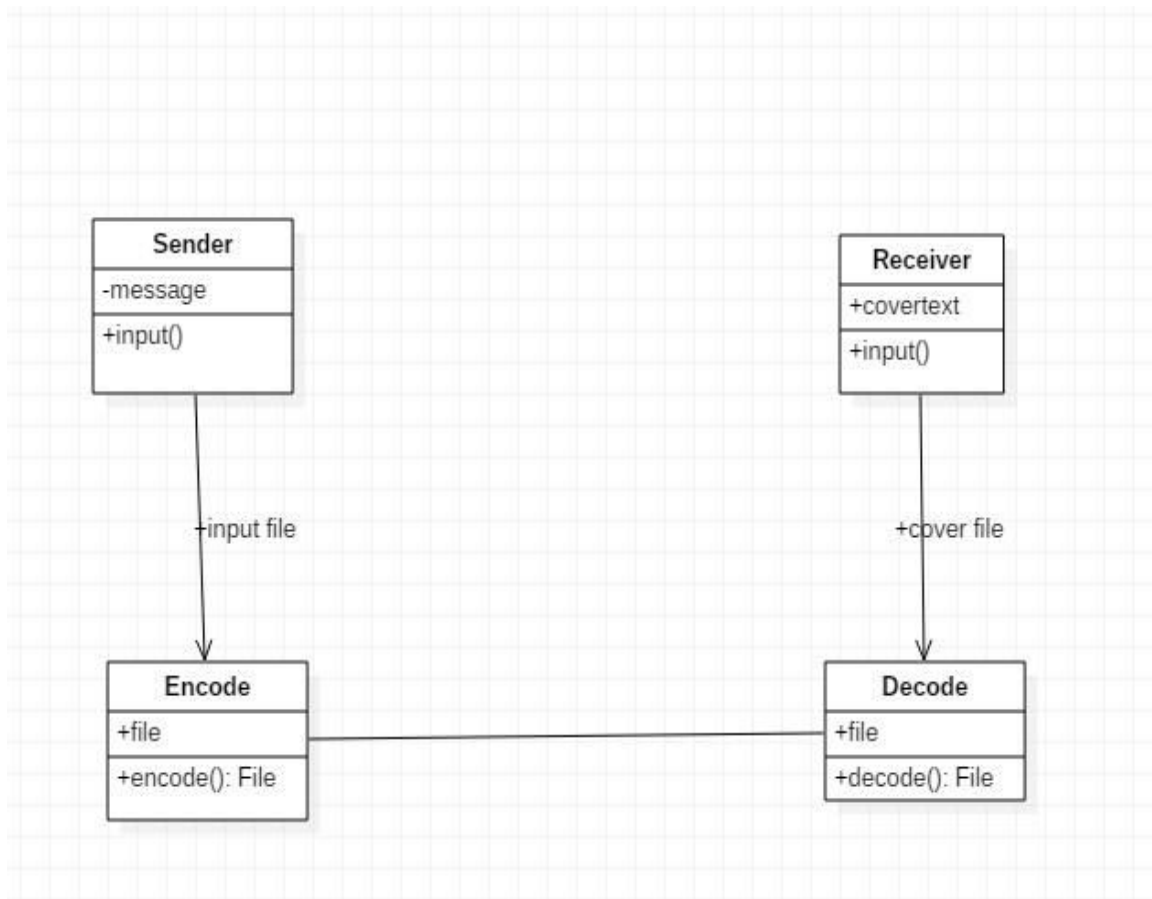


Fig 36 Class Diagram: Text Steganography project

5.4.8.2 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 functions 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 used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). 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, and class diagram is a structure diagram.

Use case diagrams are in fact twofold - they are both behavior diagrams, because they describe behavior of the system, and they are also structure diagrams - as a special case of class diagrams

where classifiers are restricted to be either actors or use cases related to each other with associations.

[UML 2.5 FTF - Beta 1] 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.

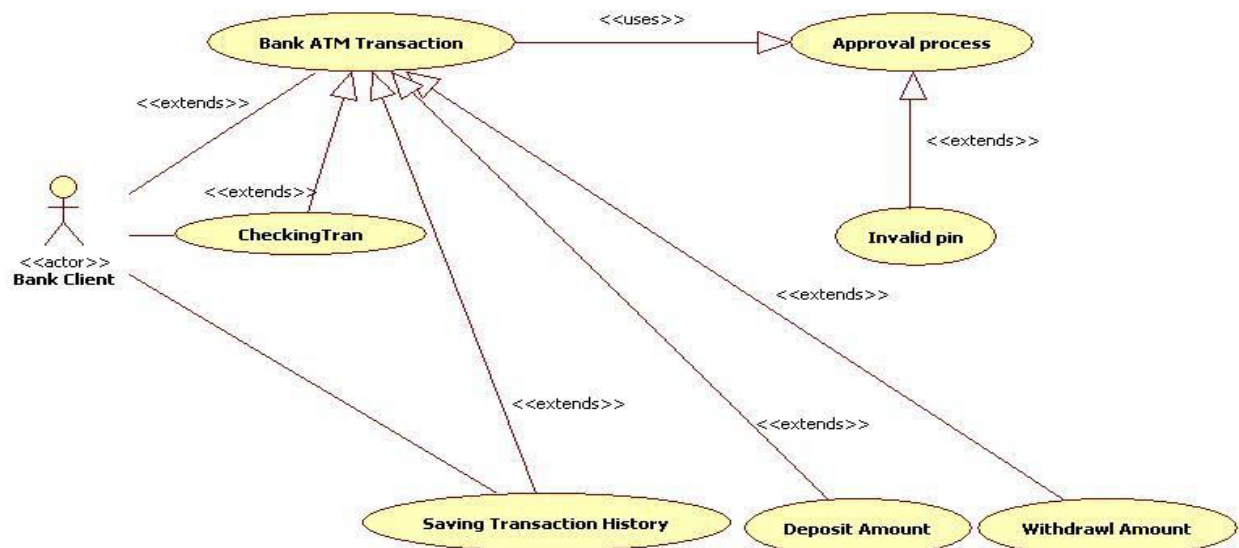


Fig 37: ATM machine Usecase diagram

5.4.8.2.1 USECASE DIAGRAM FOR PROJECT

Considering the above reference Fig 37 Use case diagram for the project consist of two actors i.e., Sender and Receiver assigned to their respective actions in which the file is taken as input from the sender side and further the cover text file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

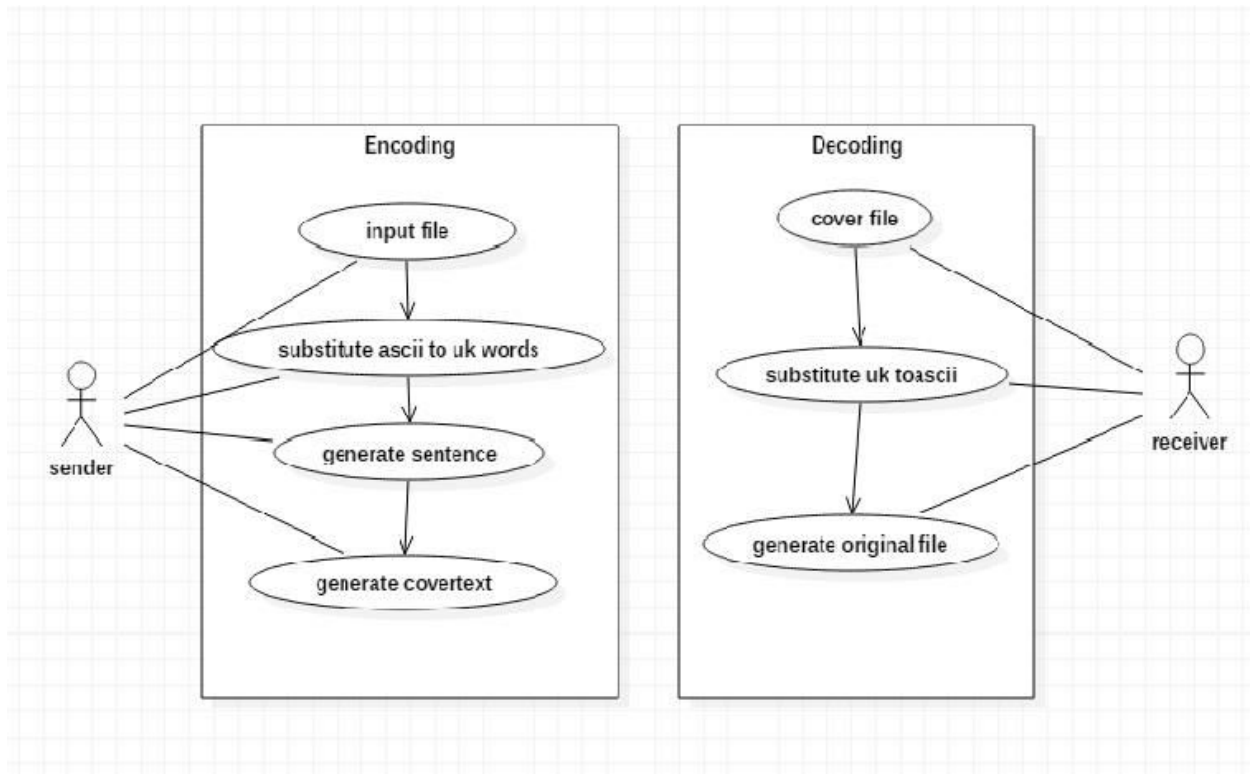


Figure 38: USECASE DIAGRAM: Text Steganography tool

5.4.8.3 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.

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.

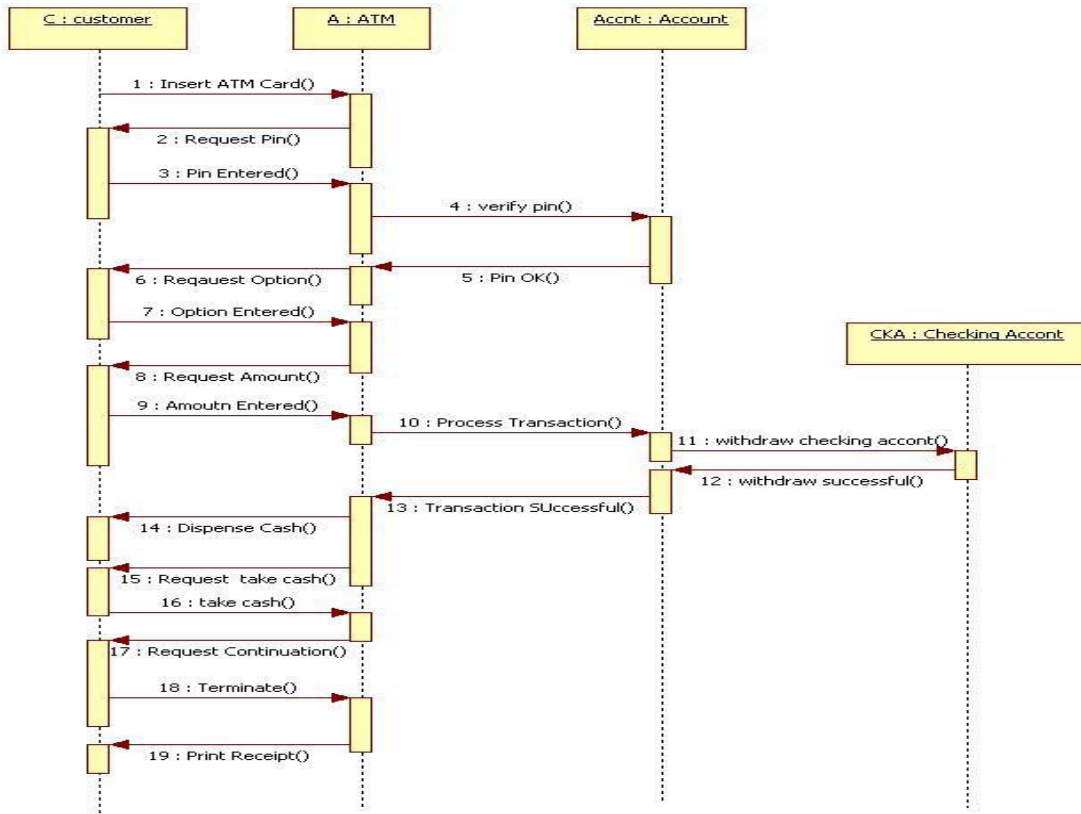


Fig 39: ATM machine Sequence diagram

5.4.8.3.1 SEQUENCE DIAGRAM FOR PROJECT

Considering the above reference Fig 39 Sequence diagram for the project consist of 4 objects in which the file is taken as input from the sender side and further the covertext file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

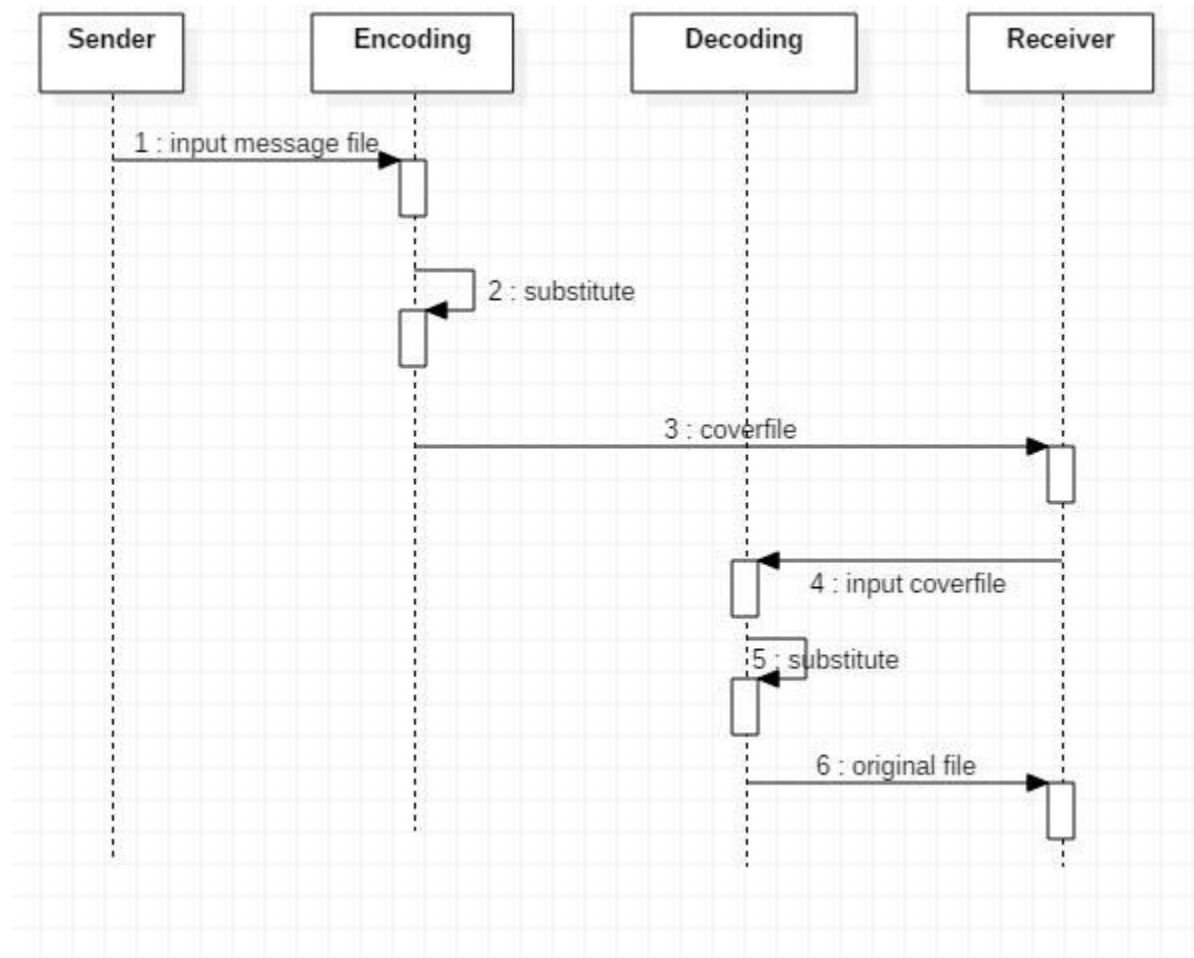


Fig 40: SEQUENCE DIAGRAM: Text Steganography tool

5.4.8.4 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.

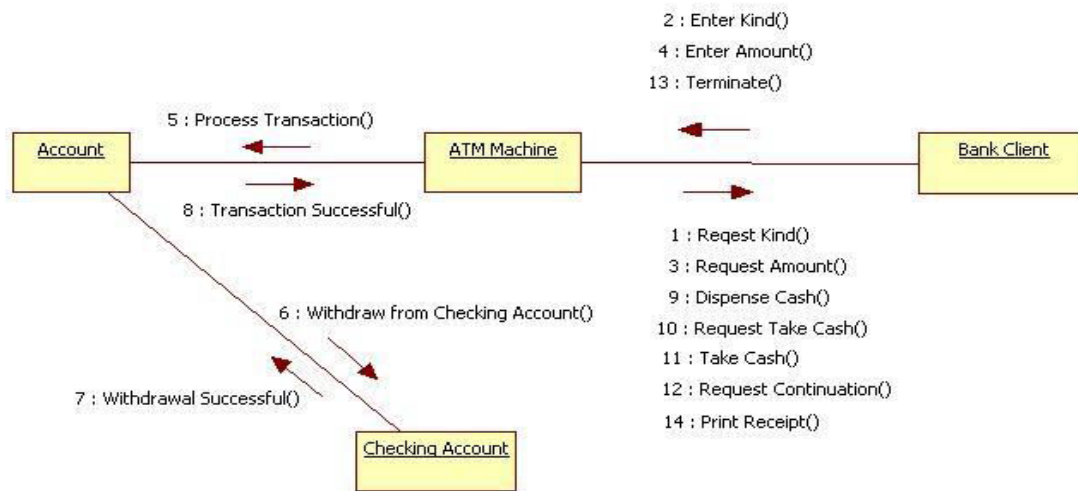


Fig 41: ATM machine Collaboration diagram

5.4.8.4.1 COLLABORATION DIAGRAM FOR PROJECT

Considering the above reference Fig 41 Collaboration diagram for the project consist of 4 units in which the file is taken as input from the sender side and further the covertext file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

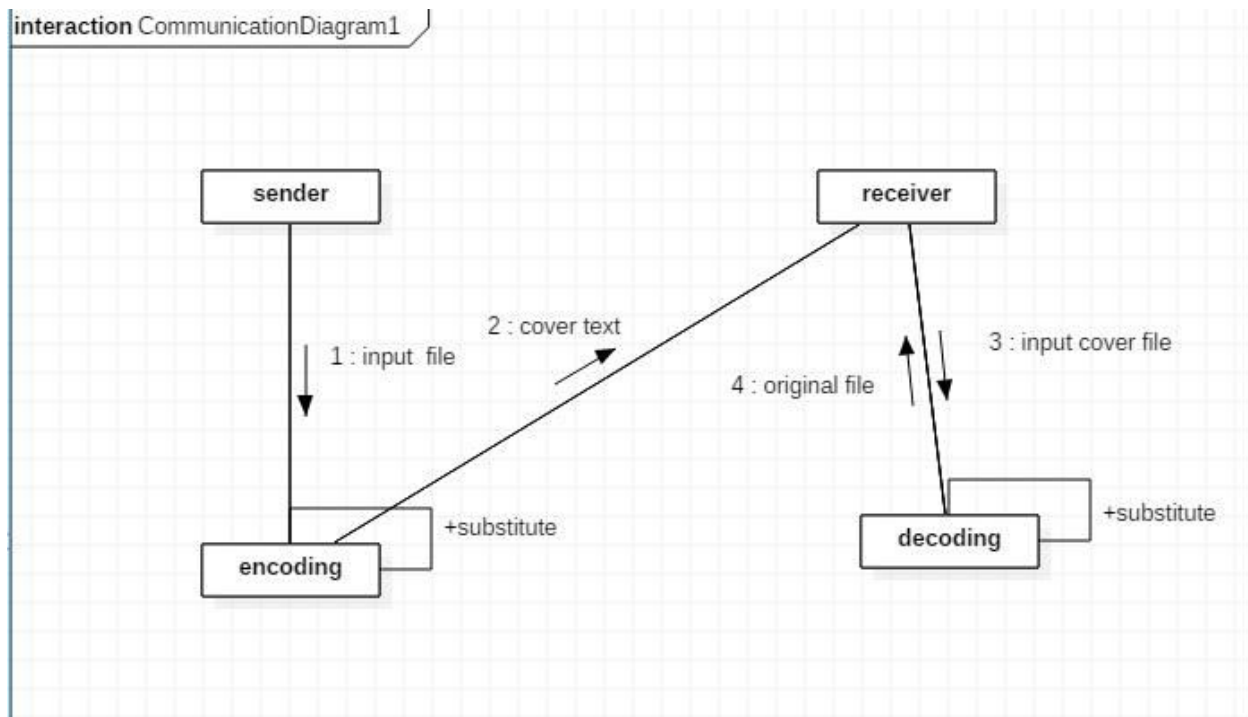


Fig 42: COLLABORATION DIAGRAM: Text Steganography tool

5.4.8.5 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.

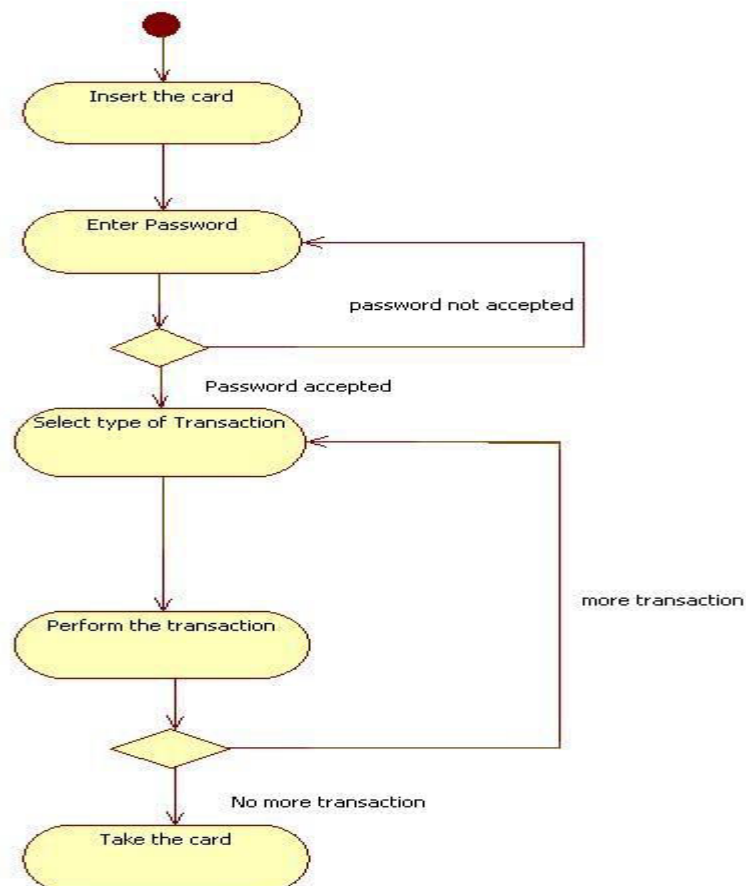


Fig 43: ATM machine Activity diagram

5.4.8.5.1 ACITIVITY DIAGRAM FOR PROJECT

Considering the above reference Fig 43 Activity diagram for the project consist of points that specifies starting and ending of project in which the file is taken as input from the sender side and further the cover text file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

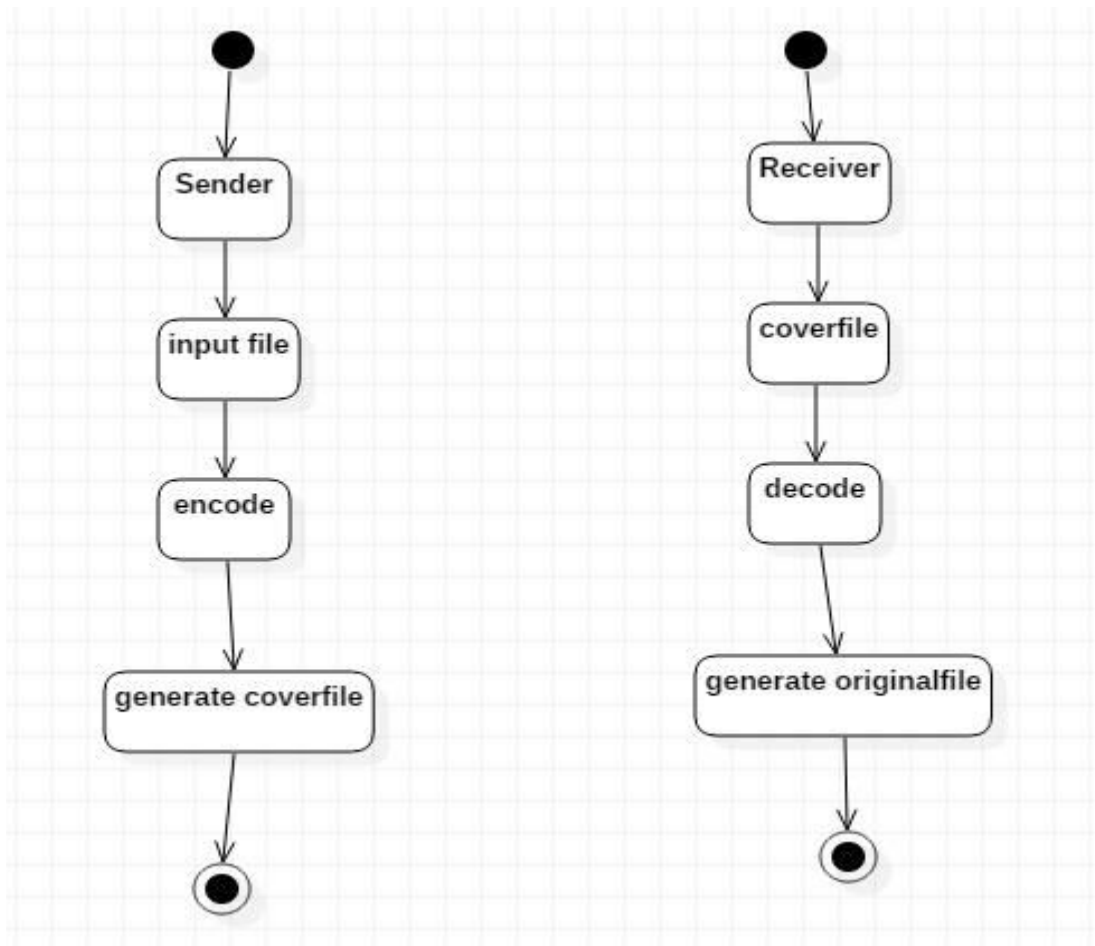


Fig 44: ACTIVITY DIAGRAM: Text Steganography tool

5.4.8.6 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 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 variables 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.

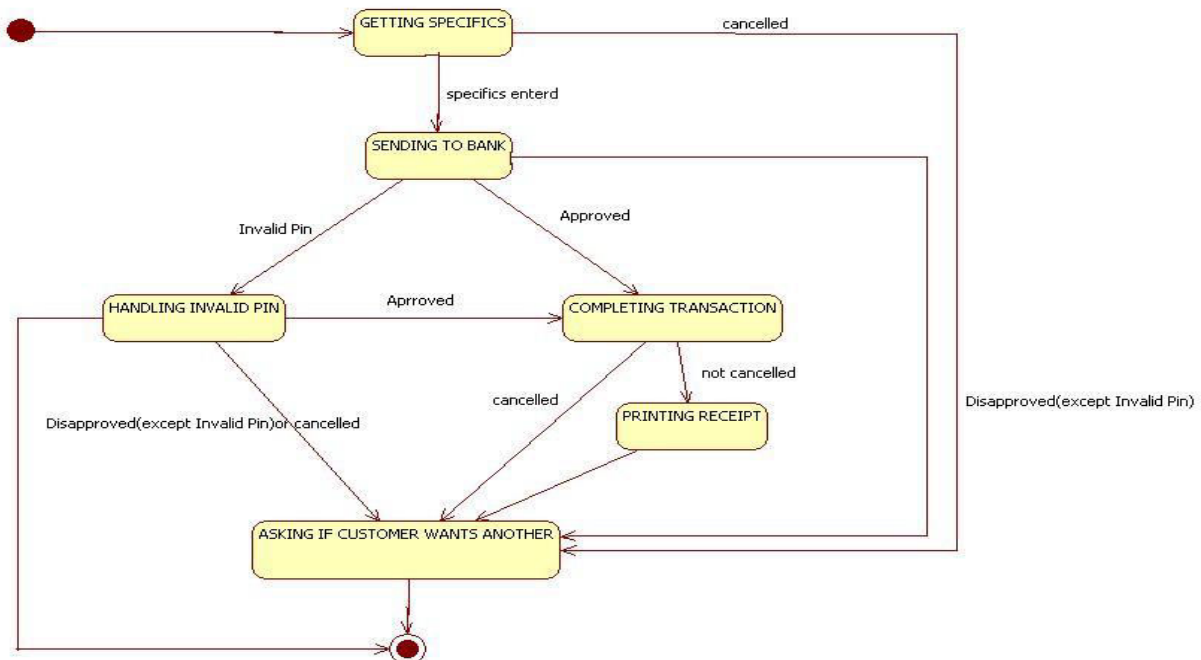


Fig 45: ATM machine State diagram

5.4.8.6.1 STATECHART DIAGRAM FOR PROJECT

Considering the above reference Fig 45 State chart diagram for the project consist of initial point and final point for the project flow in which the file is taken as input from the sender side and further the cover text file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

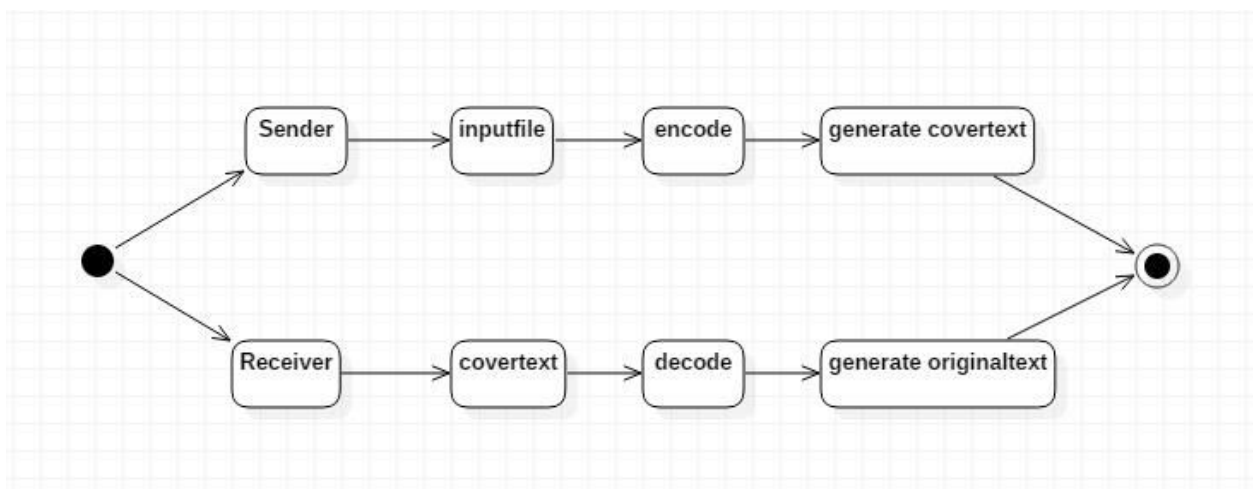


Figure 46: STATECHART DIAGRAM: Text Steganography tool

5.4.8.7 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.

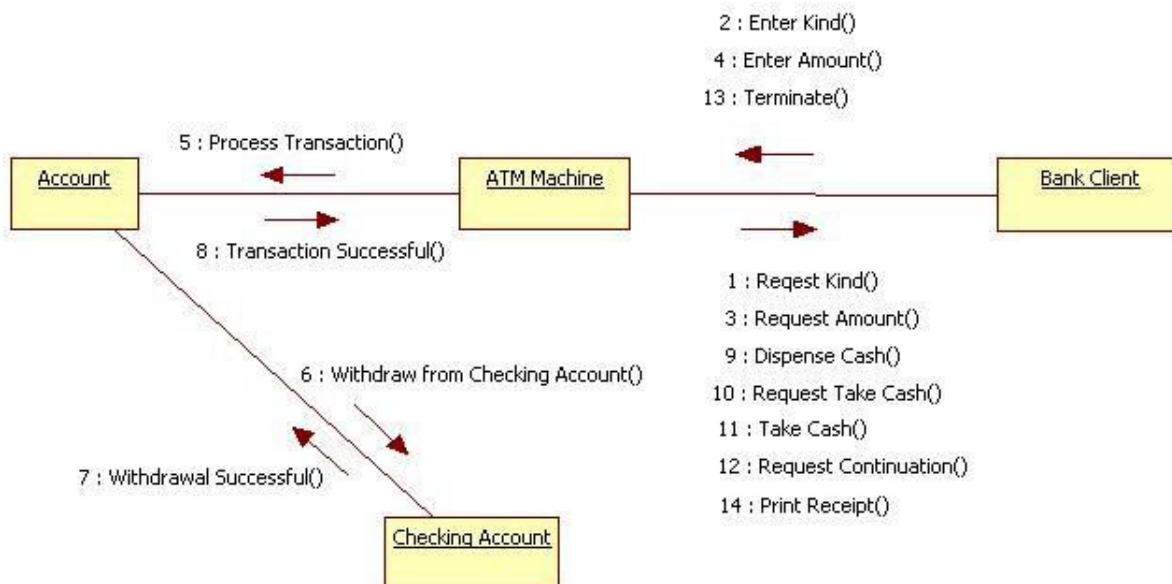


Fig 47: ATM machine Component diagram

5.4.8.7.1 COMPONENT DIAGRAM FOR PROJECT

Considering the above reference Fig 47 Component diagram for the project consist of 3 components i.e., sender user and admin in which the file is taken as input from the sender side

and further the cover text file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

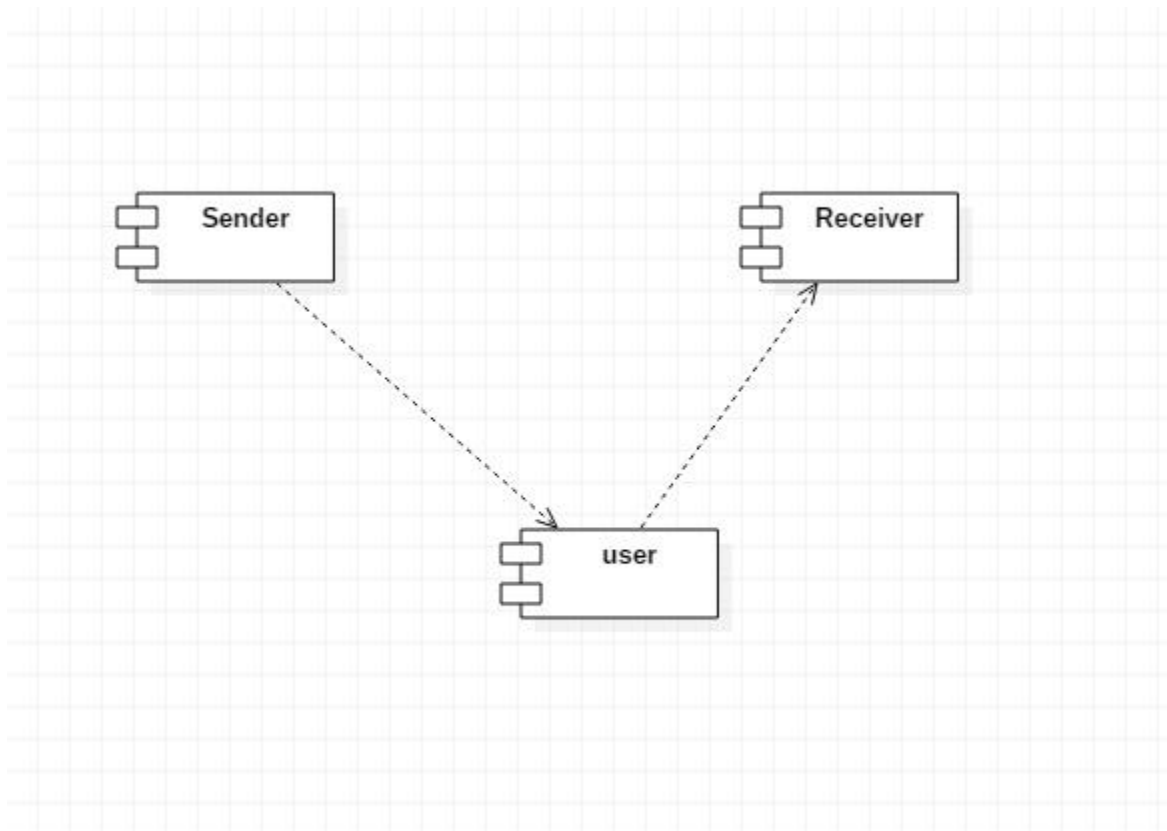


Figure 48: COMPONENT DIAGRAM: Text Steganography tool

5.4.8.8 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.

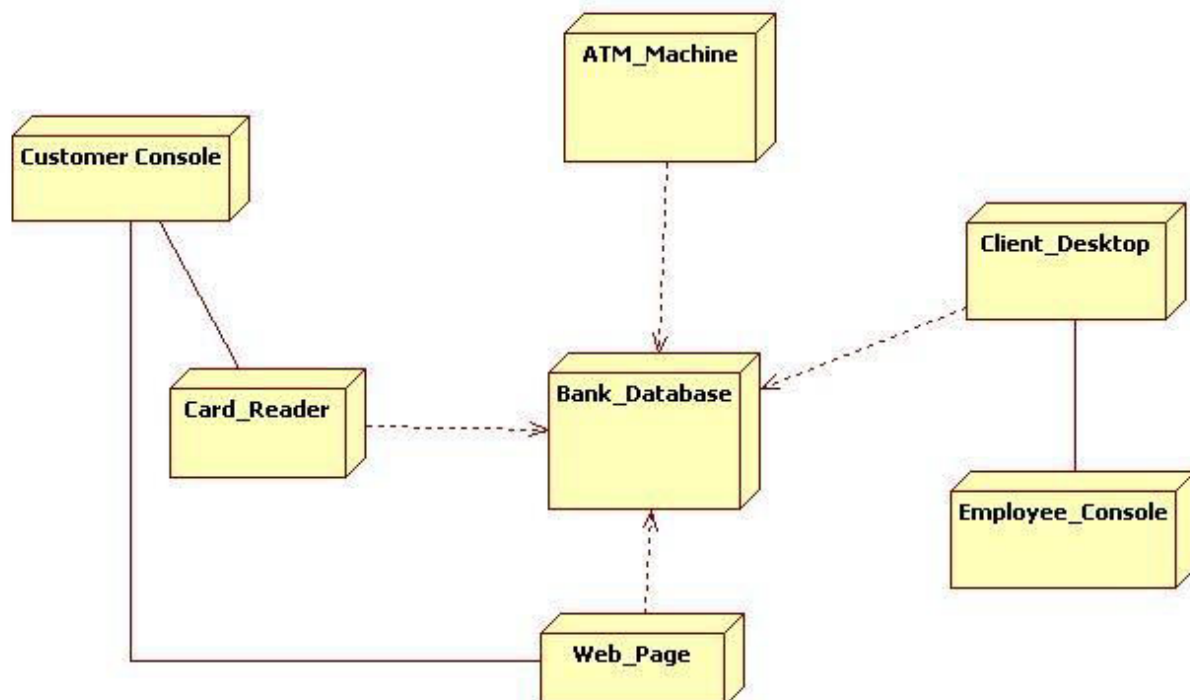


Fig 49: ATM machine Deployment diagram

5.4.8.8.1 DEPLOYMENT DIAGRAM FOR PROJECT

Considering the above reference Fig 45 Deployment diagram for the project consist of 3 nodes i.e., Sender, user and Receiver in which the file is taken as input from the sender side and further the cover text file is generated and that file is shared among the receiver and further from decoding side the original message file is generated.

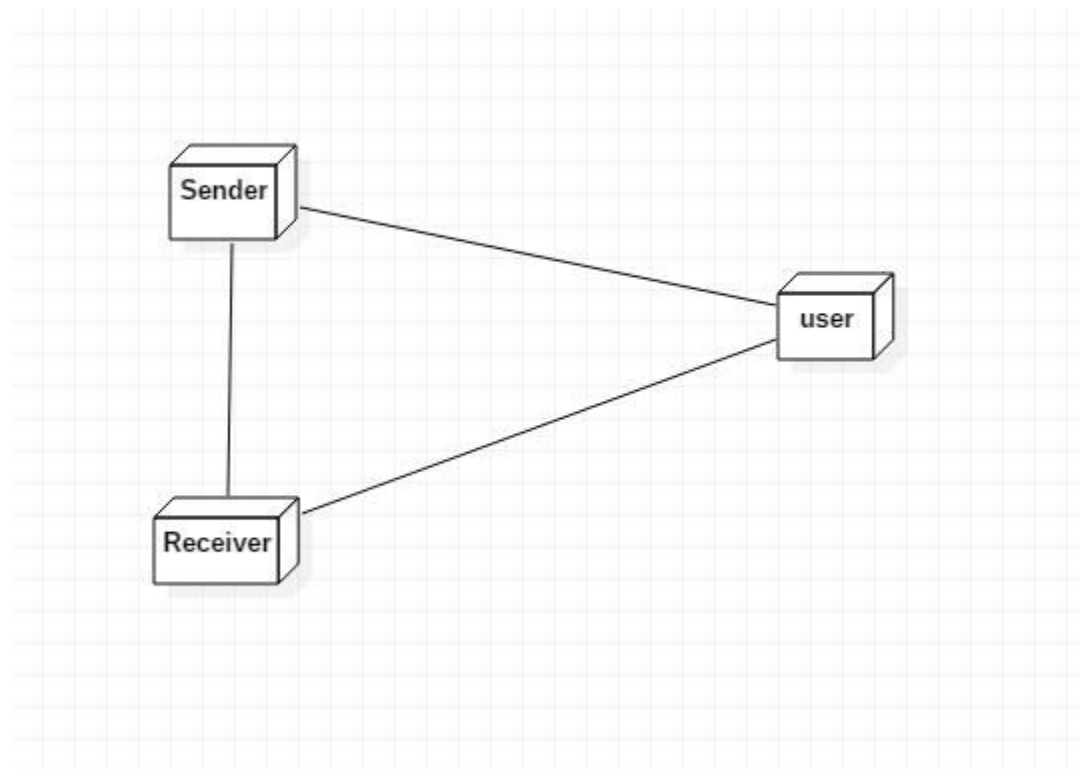


Figure 50: DEPLOYMENT DIAGRAM: Text Steganography tool

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 tougher to identify.

6.2 METHOD OF IMPLEMENTATION:

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 JLayeredPane(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)
    {
    }
    }
    });

}

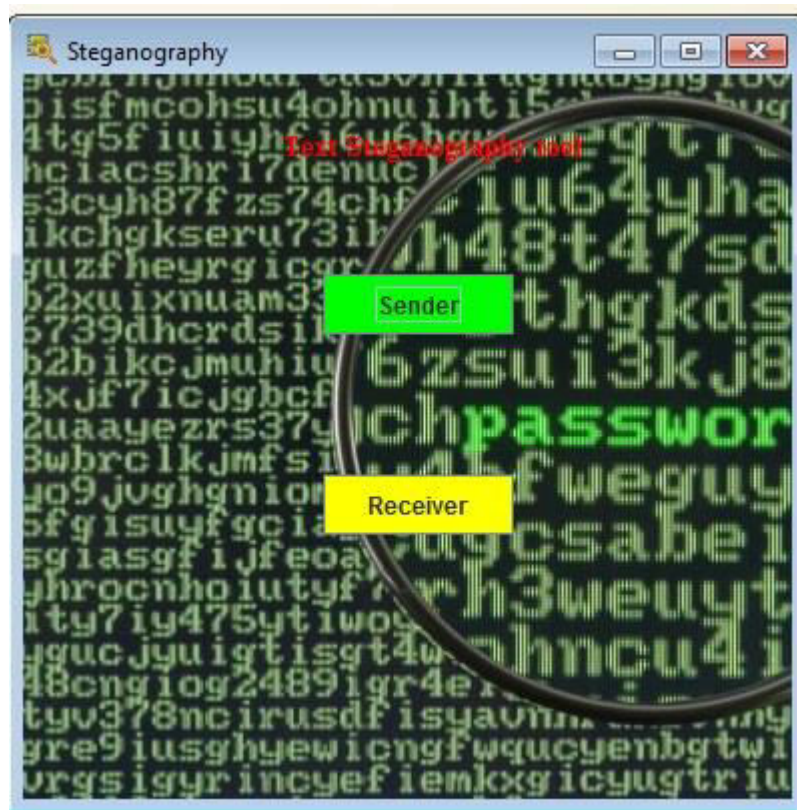
});

}

```

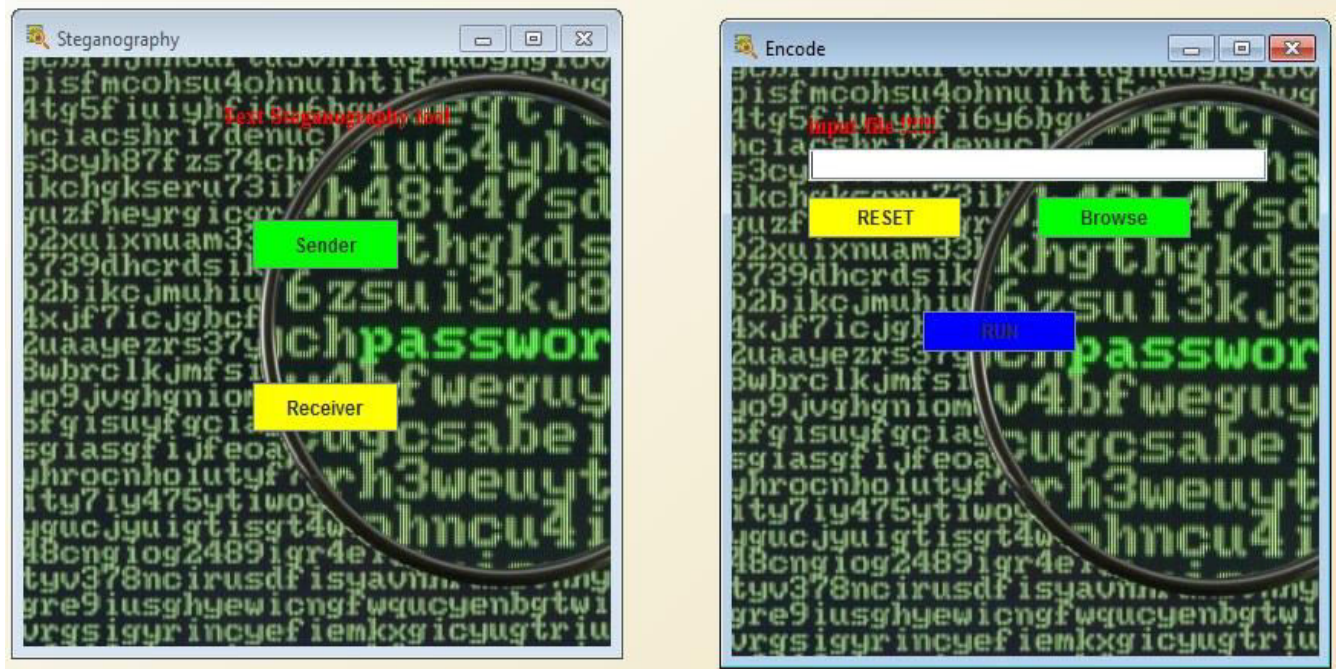
6.3 SCREEN SHORTS:

6.2.1 Main Page



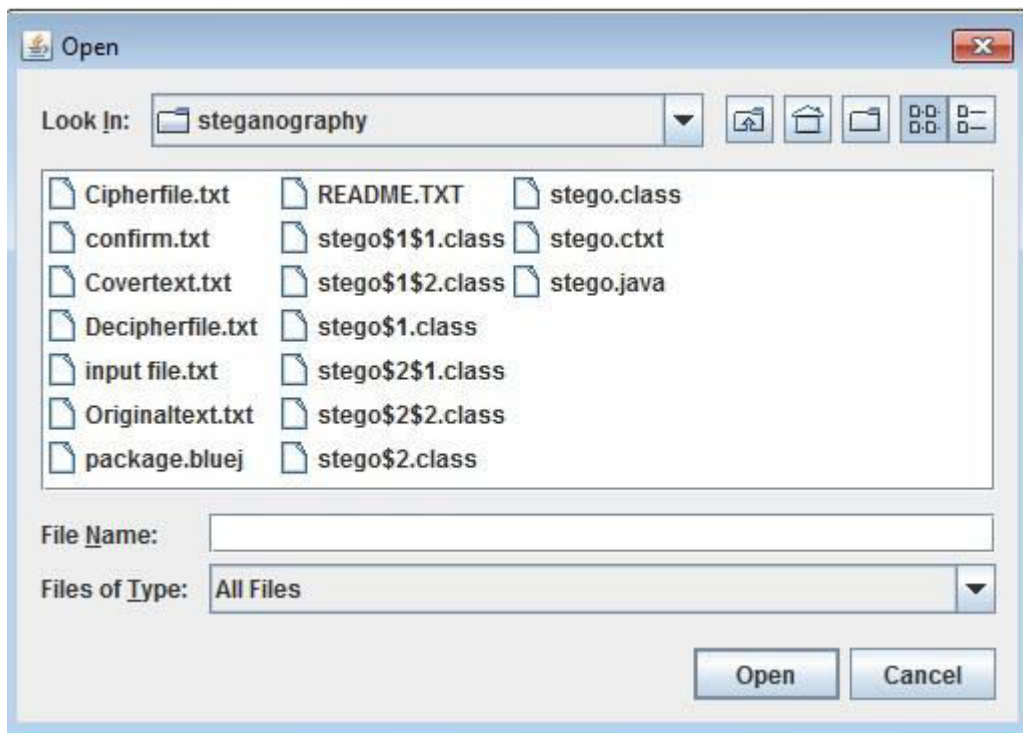
Screen 1: Main Page

6.2.2 Sender Page



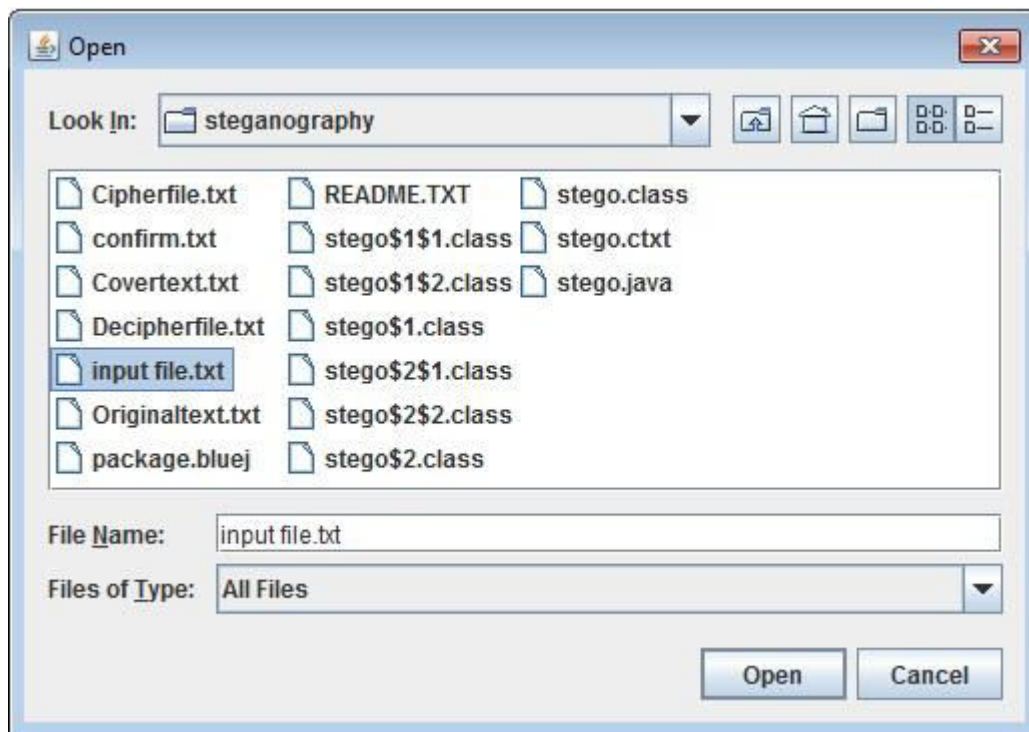
Screen 2: sender page

6.2.3 Sender file chooser



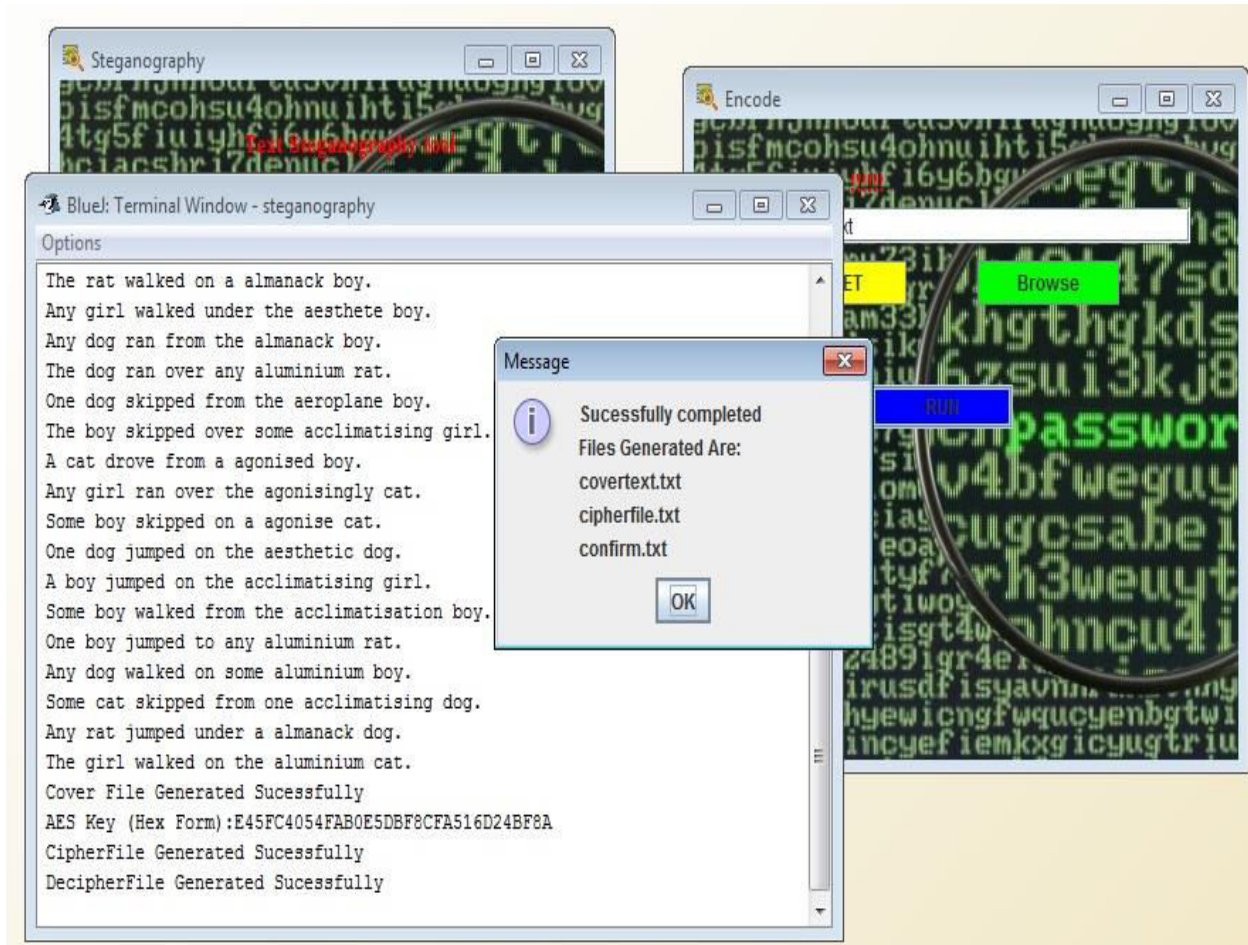
Screen 3: sender file chooser

6.2.4 Sender input file



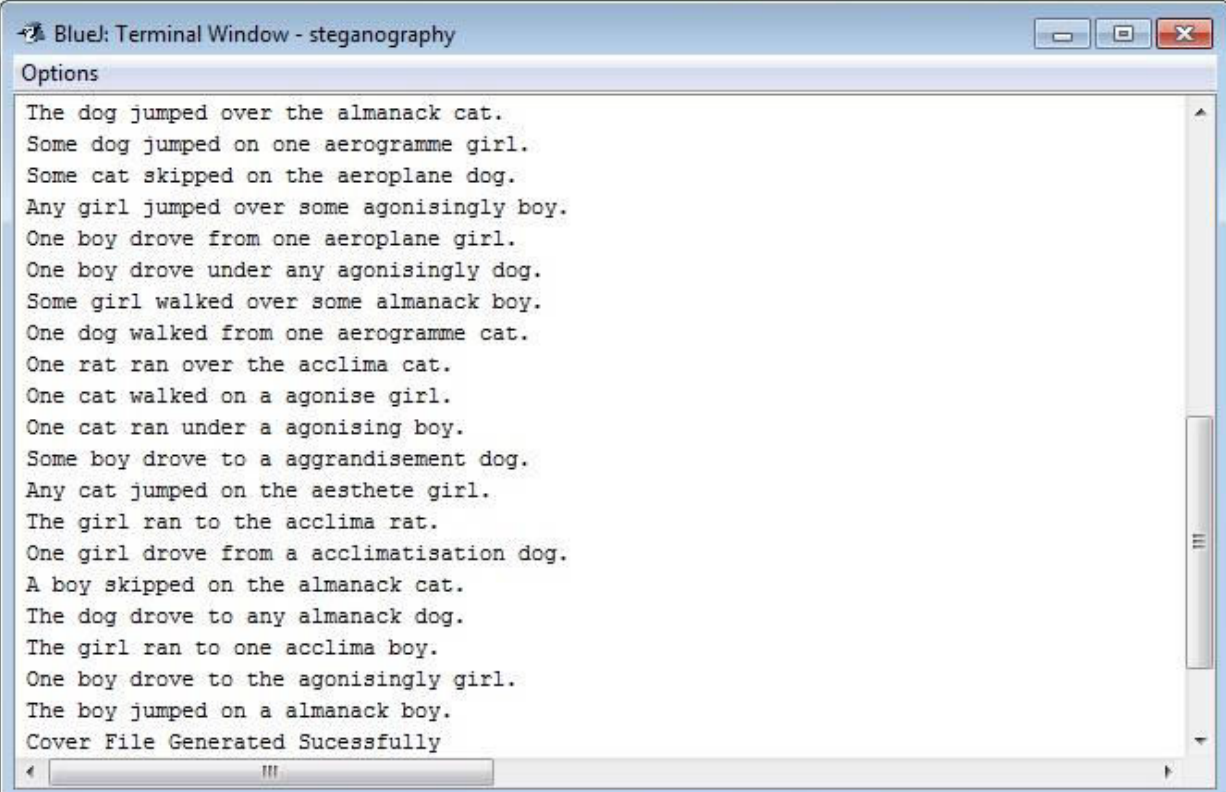
Screen 4: sender input file

6.2.4 Sender Success



Screen 5: sender success

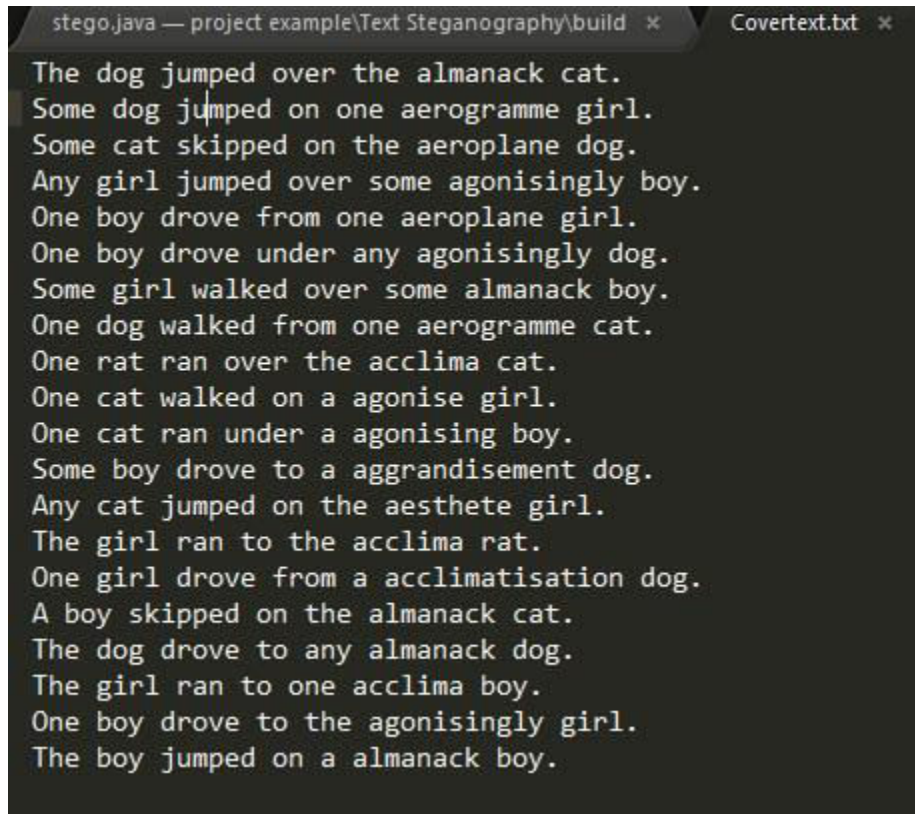
6.2.5 Sender terminal



```
BlueJ: Terminal Window - steganography
Options
The dog jumped over the almanack cat.
Some dog jumped on one aerogramme girl.
Some cat skipped on the aeroplane dog.
Any girl jumped over some agonisingly boy.
One boy drove from one aeroplane girl.
One boy drove under any agonisingly dog.
Some girl walked over some almanack boy.
One dog walked from one aerogramme cat.
One rat ran over the acclima cat.
One cat walked on a agonise girl.
One cat ran under a agonising boy.
Some boy drove to a aggrandisement dog.
Any cat jumped on the aesthete girl.
The girl ran to the acclima rat.
One girl drove from a acclimatisation dog.
A boy skipped on the almanack cat.
The dog drove to any almanack dog.
The girl ran to one acclima boy.
One boy drove to the agonisingly girl.
The boy jumped on a almanack boy.
Cover File Generated Sucessfully
```

Screen 6: sender terminal

6.2.6 Cover File



```
stego.java — project example\Text Steganography\build x Coverttext.txt x
The dog jumped over the almanack cat.
Some dog jumped on one aerogramme girl.
Some cat skipped on the aeroplane dog.
Any girl jumped over some agonisingly boy.
One boy drove from one aeroplane girl.
One boy drove under any agonisingly dog.
Some girl walked over some almanack boy.
One dog walked from one aerogramme cat.
One rat ran over the acclima cat.
One cat walked on a agonise girl.
One cat ran under a agonising boy.
Some boy drove to a aggrandisement dog.
Any cat jumped on the aesthete girl.
The girl ran to the acclima rat.
One girl drove from a acclimatisation dog.
A boy skipped on the almanack cat.
The dog drove to any almanack dog.
The girl ran to one acclima boy.
One boy drove to the agonisingly girl.
The boy jumped on a almanack boy.
```

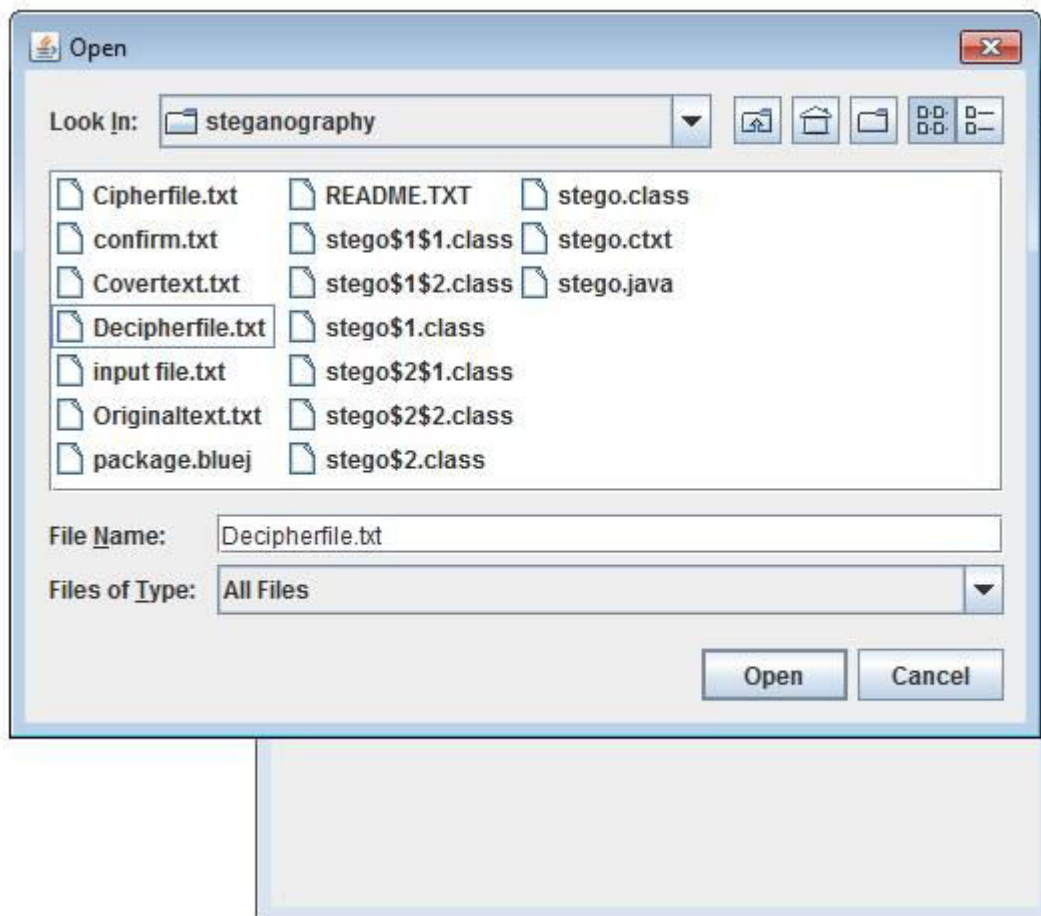
Screen 7: coverfile

6.2.7 Receiver Page



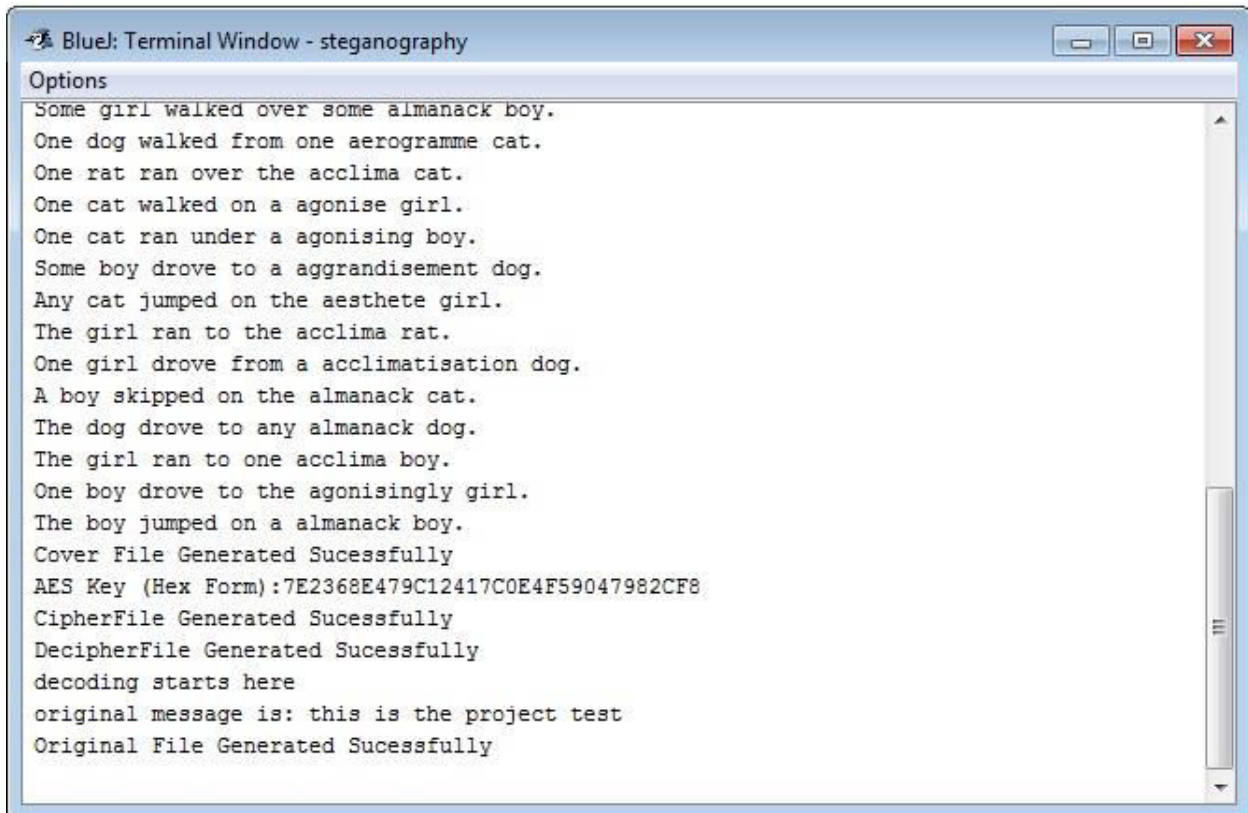
Screen 8: Receiver Page

6.2.8 Receiver file chooser



Screen 9: receiver filechooser

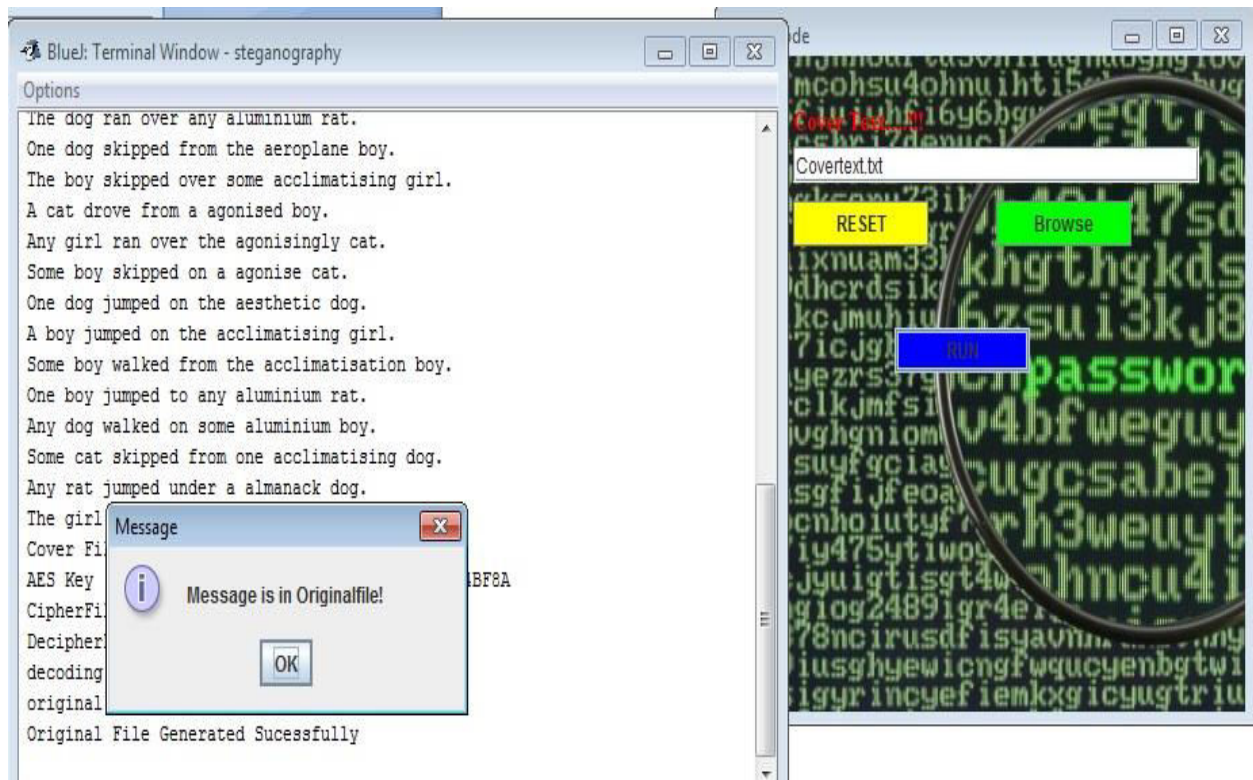
6.2.9 Receiver terminal



```
BlueJ: Terminal Window - steganography
Options
Some girl walked over some almanack boy.
One dog walked from one aerogramme cat.
One rat ran over the acclima cat.
One cat walked on a agonise girl.
One cat ran under a agonising boy.
Some boy drove to a aggrandisement dog.
Any cat jumped on the aesthete girl.
The girl ran to the acclima rat.
One girl drove from a acclimatisation dog.
A boy skipped on the almanack cat.
The dog drove to any almanack dog.
The girl ran to one acclima boy.
One boy drove to the agonisingly girl.
The boy jumped on a almanack boy.
Cover File Generated Sucessfully
AES Key (Hex Form):7E2368E479C12417C0E4F59047982CF8
CipherFile Generated Sucessfully
DecipherFile Generated Sucessfully
decoding starts here
original message is: this is the project test
Original File Generated Sucessfully
```

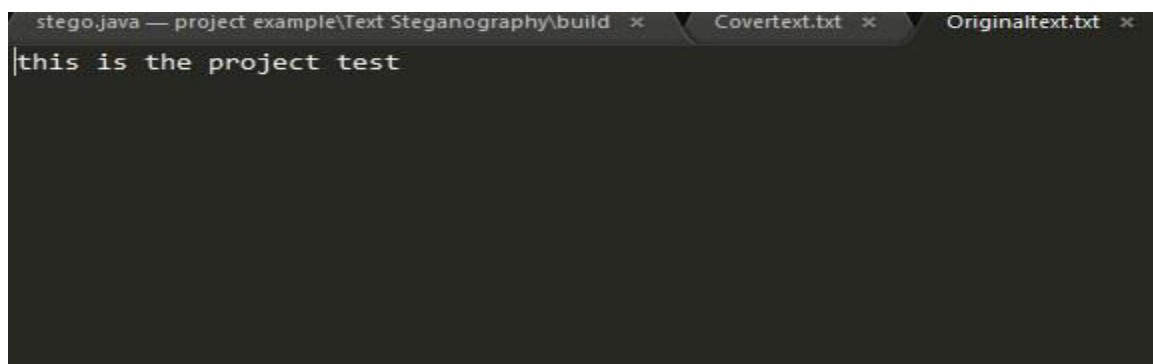
Screen 10: receiver terminal

6.2.10 Receiver success



Screen 11: receiver success

6.2.2 Original file



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.

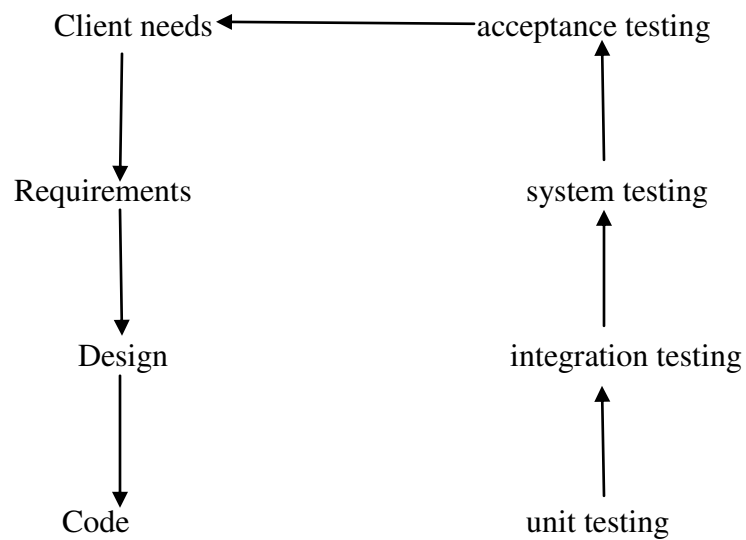
Levels of Testing: In order to uncover present in different phases we have the concept of levels of testing.

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



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



Fig 50 black box testing

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 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 project can be further extended by considering another set of words that are spelt differently and also can be extended using different languages. In this project only limited UK and US differently spelt words were used and this can be further extended to more number of words and can be framed English sentence with appropriate grammar and make it even harder to decode it.

REFERENCES

- [1] Java Documentation Webpage. <https://docs.oracle.com/en/java/>.
- [2] Roger S Pressman. Software Engineering: A Practitioner's Approach. 2010.
- [3] Visual Paradigm International. Visual Paradigm for UML 10.1 Community Edition. <http://www.visual-paradigm.com>.
- [4] Stack overflow webpage. <https://stackoverflow.com/>.
- [5] IEEE reference documents webpage. <http://ieeexplore.ieee.org/document/4494159/>.
- [6] <http://www.geeksforgeeks.org/>

TEXT STEGANOGRAPHY BY CHANGING WORD SPELLINGS

Project report submitted to J.N.T.U. Hyderabad
in partial fulfillment of the requirement for the award of degree

“Bachelor of Technology”

In

COMPUTER SCIENCE & ENGINEERING

By

P. KAVYA

14PQ1A0584

G.BHARGAV SAI

14PQ1A0554

K. SURYA

14PQ1A0564

Guide

MR. K. ADITHYA KUMAR

(Assistant professor)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANURAG COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi & Affiliated to JNTU-HYD)

AUSHAPUR (V), GHATKESAR (M), R.R.DIST, T.S.501301

July, 2017

Department of Computer Science and Engineering

CERTIFICATE

Date: / /2017

This is to certify that the project report titled **“TEXT STEGANOGRAPHY BY CHANGING WORD SPELLINGS”** submitted by **P. KAVYA** bearing **14PQA0584**, **G. BHARGAV SAI** bearing **14PQ1A0554** and **K. SURYA** bearing **14PQA0564** to the Department of Computer Science and Engineering, for the partial fulfillment of the requirement for the award of degree **“Bachelor of Technology in Computer Science & Engineering”**, by Jawaharlal Nehru Technological university, Hyderabad is a record of bonafide work carried out by his/her under our guidance and supervision in 2016-2017.

Guide

Head of Department

External Examiner

Principal

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project report.

We express our sincere gratitude to our project guide **Mr.K.Adithya Kumar** of Computer Science and Engineering, Anurag College of Engineering, for his/her precious suggestions, motivation and co-operation for the successful completion of this project report.

It is our privilege and pleasure to express our profound sense of gratitude and indebtedness to our Head of Department **Mr.B.R Srinivasa Rao**, Department of Computer Science and Engineering, Anurag College of Engineering, for his guidance, cogent discussion and encouragement throughout this project work.

We would like to express our deep sense of gratitude to **Dr.R.Anupama, Academic Coordinator**, Anurag College of Engineering, for her tremendous support, encouragement and inspiration

We express our sincere thanks to **Dr. G. Laxminarayana, Principal**, Anurag College of Engineering, for his encouragement and constant help.

We also thank the management, for providing us necessary facilities to carry out the project work at the organization.

We take this opportunity to thank all our lecturers and technical staff who have directly or indirectly helped our project. We pay our respects and love to our parents and all other family members for their love and encouragement throughout our career. Last but not the least we express our thanks to our friends for their cooperation and support

P. KAVYA 14PQ1A0584

G. BHARGAV SAI 14PQ1A0554

K. SURYA 14PQ1A0542

DECLARATION

We declare that this written submission represents my ideas in my own words and where others ideas or words have been included. I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

We hereby declare that the results embodied in this project report titled “**TEXT STEGANOGRAPHY BY CHANGING WORD SPELLINGS**” requirements for the award of degree are obtained by us from our project. We have not submitted this project report to any other University/Institute for the award of any degree/Diploma.

P. KAVYA 14PQ1A0584

G. BHARGAV SAI 14PQ1A0554

K. SURYA 14PQ1A0564

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 be seen by the third party so the research has been started on it and invented some methods in its behalf. 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 in a 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 techniques to implement 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 are 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.

LIST OF FIGURES

FIG NO	TITLE	PAGE NO
1	Secure transmission techniques	1
2	Cryptography to Steganography	5
3	Steganography Working Model	9
4	Types of Steganography	9
5	Types of Text Steganography	11
6	HTML file	17
7	Font Types in MS Word	18
8	Font table	19
9	Sender	23
10	Receiver	24
11	System Architecture	26
12	ER Diagram	28
13	Structural things	29
14	Use case structure	30
15	Use case structure	30
16	Interaction	30
17	Dependency	31
18	Association	31
19	Generalization	31
20	Realization	32
21	Actor	32
22	Object	33
23	Unit	33
24	Separator	33
25	Group	34
26	Action	34
27	Asynchronous Message	34
28	Block	34
29	Call message	34
30	Create message	35
31	Diagram name	35

FIG NO	TITLE	PAGE NO
32	Message	35
33	Return Message	36
34	ATM class diagram	38
35	Class diagram	39
36	ATM Use case diagram	40
37	Use case diagram Project	41
38	ATM Sequence diagram	42
39	Sequence Diagram	43
40	ATM Collaboration diagram	44
41	Collaboration diagram	45
42	ATM machine Activity diagram	46
43	Activity diagram	47
44	ATM machine State diagram	48
45	State diagram	49
46	ATM machine Component diagram	50
47	Component diagram	51
48	ATM machine Deployment diagram	52
49	Deployment diagram	53
50	Black box testing	78

LIST OF SCREENS

SCREEN NO	TITLE	PAGE NO
1	Main page	64
2	sender page	65
3	sender file chooser	66
4	sender input file	67
5	sender success	68
6	sender terminal	69
7	Cover file	70
8	Receiver Page	71
9	receiver file chooser	72
10	receiver terminal	73
11	receiver success	74
12	Original txt	74

LIST OF ACRONYMS

S NO	ACRONYM	FULL FORMS
1	IS	Information Security
2	ERD	Entity-Relationship Diagram
3	UML	Unified Modeling Language
4	JDK	Java Development Kit
5	IDE	Integrated Development Environment
6	JRE	Java Runtime Environment
7	ADT	Abstract Data Type
8	API	Application Program Interface

CONTENTS

LIST OF FIGURES	I
LIST OF SCREENS	III
LIST OF ACRONYMS	VI
CHAPTER 1	
INTRODUCTION	
1.1 Introduction	01
1.2 Motivation	06
1.3 Problem Definition	06
1.4 Objective of Project	06
1.5 Organization of Project	06
CHAPTER 2	
LITERATURE SURVEY	
2.1 Introduction	08
2.2 Literature Review	14
2.2.1 Text Steganography in SMS	14
2.2.2 Text Steganographic Algorithms for HTML	16
2.2.3 Text Steganography On Font Type in Ms-Word	17
2.3 Problem definition	20
2.4 Proposed System	20
CHAPTER 3	
REQUIREMENT SPECIFICATIONS	
3.1 Software requirement	21
3.2 Hardware requirement	21
CHAPTER 4	
MODULE DESCRIPTION	
4.1 Introduction	23
4.2 Sender	23
4.3 Receiver	24

CHAPTER 5

DESIGN AND ANALYSIS

5.1 Introduction	25
5.2 System Architecture	25
5.2 Entity Relationship Diagram	26
5.3 UML diagram	28

CHAPTER 6

IMPLEMENTATION AND RESULTS

6.1 Introduction	54
6.2 Method of Implementation	54
6.3 Screenshots	64

CHAPTER 7

TESTING & VALIDATION

7.1 Introduction	75
8.2 Design of test cases and scenarios	76

CHAPTER 8

CONCLUSION

8.1 Project conclusion	80
8.2 Future Enhancement	81

REFERENCES	82
-------------------	----

