IMAGE STEGANOGRAPHY

-with smooth image encoding and secured key protection

Rahul Yerramsetti
Email: rahulyerramsetti@rocketmail.com

ABSTRACT

Steganography is a method of hiding secret messages in a cover object while communication takes place between sender and receiver. Security of confidential information has always been a major issue from the past times to the present time. Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size. Here this application is based on steganography using image files. Image steganography is much preferred for confidentiality due to its less susceptibility to cryptanalysis compared to other files. Here in this process we use a procedure based on the logic of embedding the Secret text data bits into the least significant bit of Image pixel data array among the total of 8 bits by using the OR operation and thereafter securing it with a random translated user entered embedded password in the image. This allows in generating the stego with least possible change/noise and doubt with further added security to the image using the translations of the bits. Hence this approach helps in smooth secured way of information exchange between multiple parties using cryptography.

CONTENTS

1. INTRODUCTION

Our goal is to build a simple application that is able to send and receive confidential messages embedded inside an image which is further protected with a random translated user entered embedded password.

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word steganography combines the Greek words steganos (στεγανός), meaning "covered, concealed, or protected", and graphein (γράφειν) meaning "writing".

The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages—no matter how unbreakable—arouse interest, and may in themselves be incriminating in countries where encryption is illegal. Thus, whereas cryptography is the practice of protecting the contents of a message alone, steganography is concerned with concealing the fact that a secret message is being sent, as well as concealing the contents of the message.

Here in this application/process, we implemented a procedure based on the logic of embedding the Secret text data bits into the least significant bit of Image pixel data array among the total of 8 bits by using the OR operation and thereafter securing it with a random translated user entered embedded password in the image. This allows in generating the stego with least possible change/noise and doubt (during Cryptanalysis or manual spottable difference) with further added security to the image using the translations of the bits. This is a java based application designed for the Microsoft Windows family.

This project helps in sending the information with added security and confidentiality of the user with the help of images. The project is alone concerned with the generation of the stego object with added security and selection of the communication channel of the image to the receiver is the sender’s wish.

2. SYSTEM STUDY

2.1. EXISTING SYSTEM

The traditional system of steganography is based encoding of the hidden text string data bits with entire bits of a Image pixels (which leads to rapid changes in the actual colour & generated colour pixels after embedding) without any further security features which causes:

High Noise in the stego generated

Least security offered during the cryptanalysis

Less secured Decryption (as anyone with the software can open it)

2.2. PROPOSED SYSTEM

The proposed system is based on encoding the hidden text data bits to the image pixel data bits only at their each least significant bits of a pixels(which leads to negligible changes in the actual colour & generated colour pixels after embedding) along with security provided with a random translated user entered embedded password.

It has the benefits as follows compared to that of the traditional system:

Less Noise generation

Security and confidentiality of the image at cryptanalysis

More secured Decryption (as the person only who knows the password can only decrypt the file)

2.3. FEASABILITY STUDY

This application has the probable efficient use in:

Building Colour printers so as to identify them with the help of random colour dots in the image

Especially by Intelligence Services and military organizations for communication with their agents or among officials in those organizations.

Research Organizations for secret communication about their research.

2.4. SDLC MEYHODOLOGY

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.

2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

• At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.

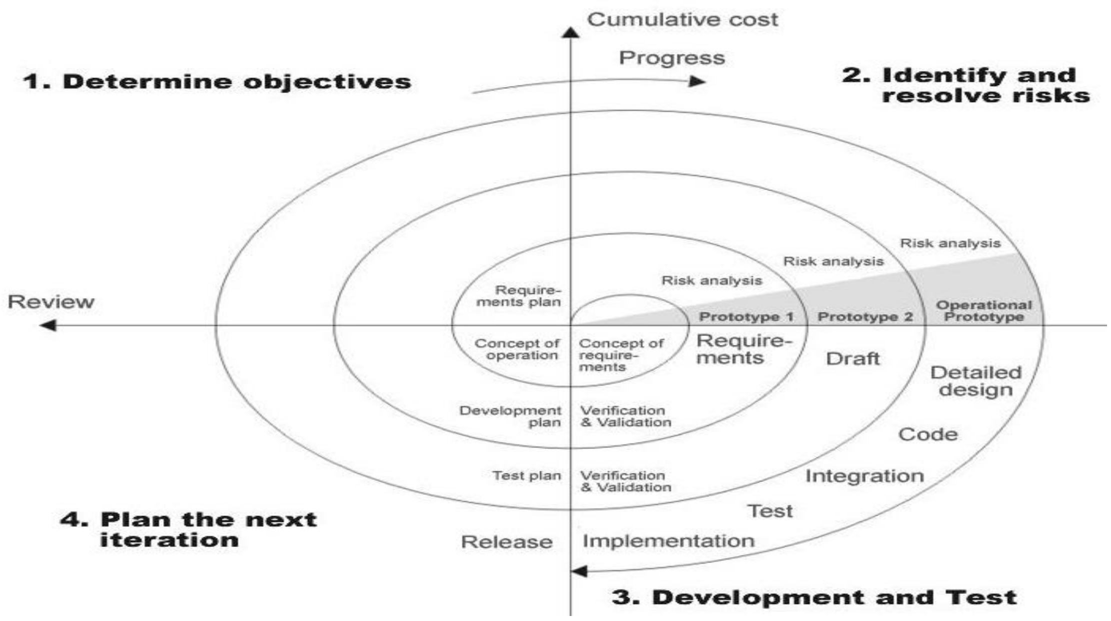
• The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.

• The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.

• The final system is constructed, based on the refined prototype.

• The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

The following diagram shows how a spiral model acts like:



This Project has gone through 5 Such Loops for the completion as follows:

Encryption Module > Decryption Module > Integration > Password addition > Logic Improvements

2.5. SYSTEM SPECIFICATIONS

2.5.1. SOFTWARE REQUIREMENTS

- Software Requirements:
- Front End: Java Swings
- Back End: Java SE8
- Platform: JDK
- OS: Windows XP and family
- IDE: Eclipse Luna

2.5.2. HARDWARE REQUIREMENTS

- Intel Pentium IV and above Processor
- 150 GB Hard Disk
- 2GB RAM
- VGA Monitor

2.5.3. FUNCTIONAL REQUIREMENTS

- Sender should open the application
- Sender should enter his secret text he intends to send and press ENTER TEXT button.
- Sender should browse a picture on which he intends to merge his text with by pressing the Browse button.
- Sender should then enter the Password with a Maximum length of 10 characters and minimum of 2 characters and press ENTER PASSWORD button.
- Sender should press the Generate Stego button to generate the Stego object by giving the name of the image file in a dialog box that appears asking to enter the file name.
- The Stego is successfully generated at the sender site.
- The sender needs to now send his generated Stego object as an image file to the Receiver with his wish of choosing his own secured communication channel (like E-Mail, CD, or other means) as he desires.
- The receiver after receiving the file (stego object) opens the application
- The receiver then browses the stego object and inputs it into the application
- The receiver Enters their shared password and press ENTER PASSWORD button.
- The receiver then presses the DECRYPT button get the hidden text.
- If the password generated by the receiver is correct the secret message/information is retrieved and displayed on a dialog box otherwise a message saying wrong password is displayed.

2.5.4. NON-FUNCTIONAL REQUIREMENTS

- A Good Shared password of Max Length 10 characters and min of 2 characters
- A secured transmission channel
- User Friendly

3. SYSTEM DESIGN

3.1. OVERVIEW OF THE PROJECT

Eclipse Luna: **Eclipse** is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins, including: Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematical. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the Eclipse Public License, Eclipse SDK is free and open-source software (although it is incompatible with the GNU General Public License). It was one of the first IDEs to run under GNU Classpath and it runs without problems under IcedTea.

3.2. MODULARIZATION

There are 4 main modules in the project as follows :

3.2.1. Image Text Encoding (Encryption Backend):

It deals with collection of the input data from the frontend class and application of the encoding logic to combine the image with the secret text and generate a smooth rasterized image. It involves the process of converting the input raw image and Secret text into appropriate Byte array's so that the encoding can be done on the bits of data and thereafter It involves the encoding of the text with the image byte data only at its least significant bit position of the byte so that an image with least possible change is generated.

3.2.2. Image Text Decoding (Decryption Backend):

It deals with collection of the input data from the frontend class and application of the decoding logic to separate the embedded image with the secret text and display the text separately without disturbing the image. It involves the process of converting the input embedded raw image into appropriate Byte array stream so that the decoding can be done on the bits of data of the image and thereafter It involves the decoding of the text from the image byte data from its least significant bit position of the byte.

3.2.3. Encryption GUI:

It involves the creation of the user interface using the Java Swings for the front end of the project which helps the user in entering the software with the image and the secret text he intends to embed into it. It involves the creation and defining various action listeners to the Jcomponents used as per the criteria required and handling the flow of events in the design. It also involves the passage of the input parameter to the backend class appropriately. It also comprises of organising the layouts of the entire GUI and switching between the tabbed panes used in the design process.

3.2.4. Decryption GUI:

It involves the creation of the user interface using the Java Swings for the front end of the project which helps the user in entering the software with the embedded Image (stego) which he intends to extract the hidden secret message from it. It involves the creation and defining various action listeners to the Jcomponents used as per the criteria required and handling the flow of events in the design. It also involves the passage of the input parameter to the backend class appropriately and the display of the result.

3.3. UML DIAGRAMS

3.3.1. USECASE DIAGRAM

Use Case diagrams are one of the five diagrams in UML for modeling the dynamic aspects of systems. Use case diagrams are central to modeling the behavior of a system, a subsystem, or a class. Each one shows a set of use cases and actors and their relationships. Use case Diagrams represent the functionality of the system from a user's- point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

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.

Things in the UML

There are four kinds of things in the UML

- **Structural Things**
- **Behavioral Things**
- **Grouping Things**
- **Annotational Things**

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

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

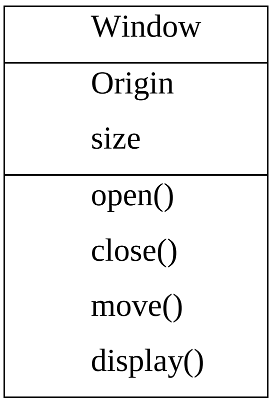


Fig: 3.1 Classes

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.

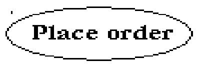


Fig: 3.2 Use Cases

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.

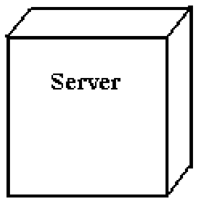


Fig: 3.3 Nodes

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

Interaction: An interaction is a behaviour 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 behaviour invoked by a message, and links (the connection between objects).

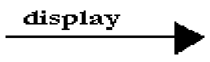


Fig: 3.4 Messages

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).

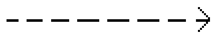


Fig: 3.5 Dependencies

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.

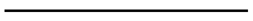


Fig: 3.6 Association

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).

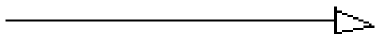


Fig: 3.7 Generalization

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

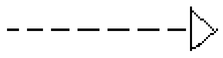


Fig: 3.8 Realization

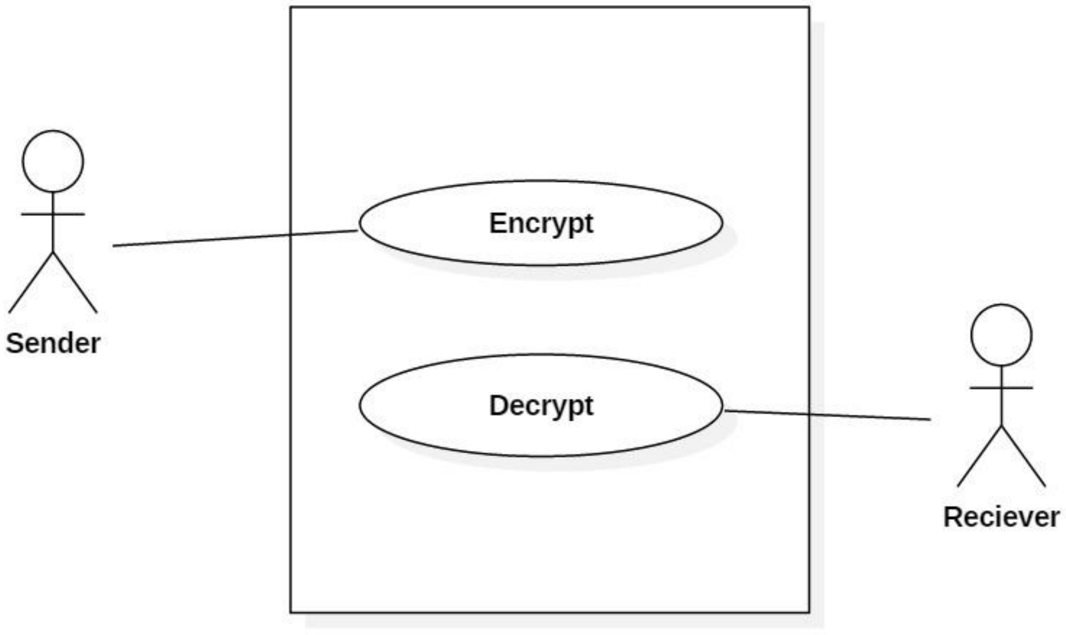


Fig: 3.9 Use Case diagram for steganography

3.3.2. CLASS DIAGRAM

Class diagrams are the most common diagrams found in modeling object-oriented systems. A class diagram shows a set of classes, interfaces, and collaborations and their relationships. These are important not only for visualizing, specifying and documenting structural models, but also for constructing executable systems through forward and reverse engineering.

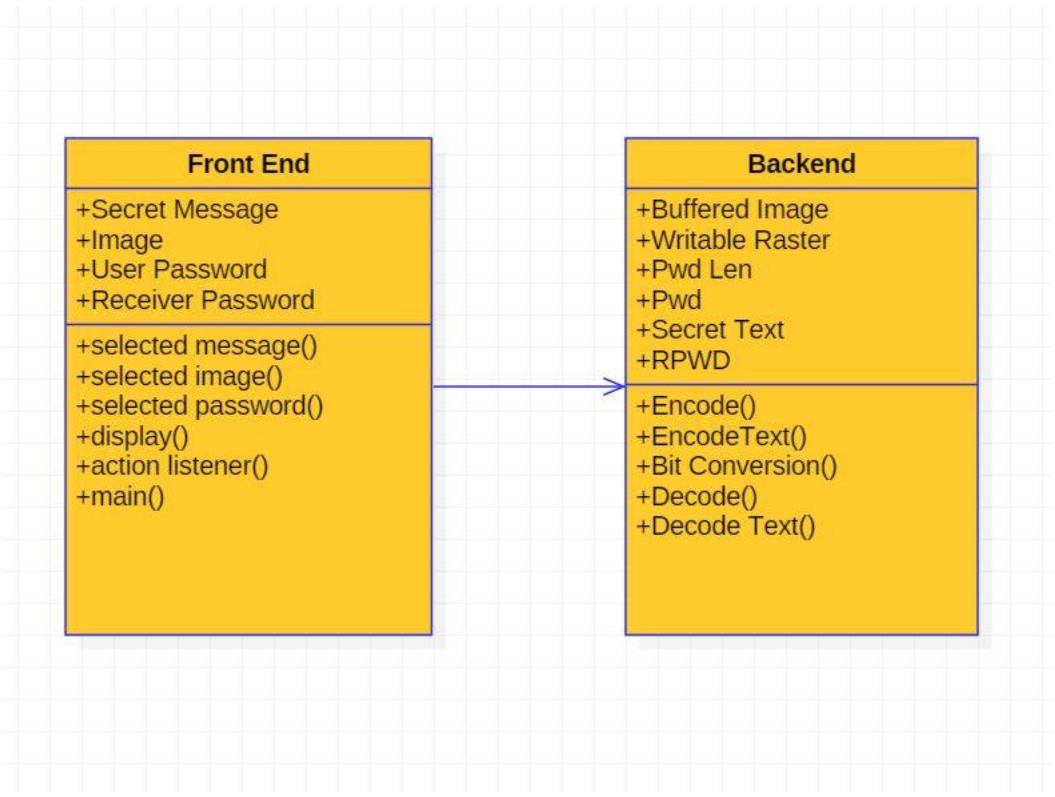


Fig3.10: Class diagram for steganography

3.3.3. SEQUENCE DIAGRAM

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages. Graphically, a sequence diagram is a table that shows objects arranged along X-axis and messages, ordered in increasing time along the Y-axis.

It contains the object lifeline that represents the existence of an object over a period of time. There is a focus of control that shows the period of time during which an object is performing an action, either directly or through a subordinate procedure.

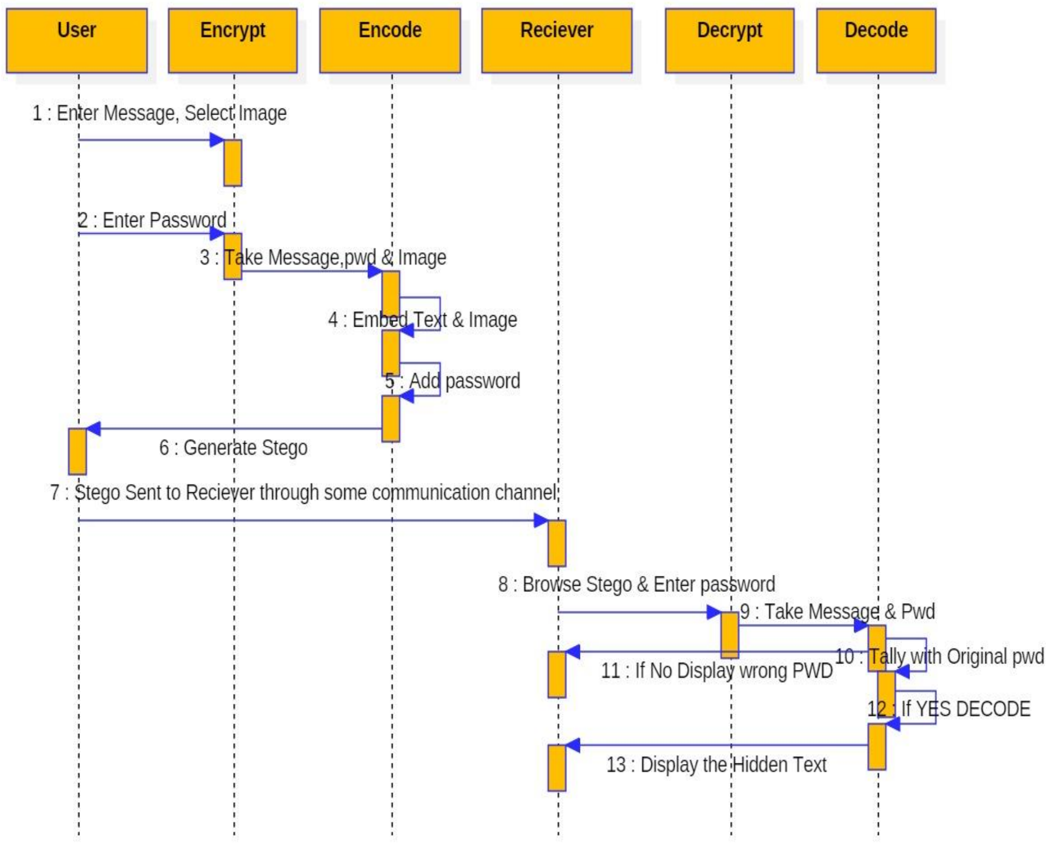

User
Encrypt
Encode
Reciever
Decrypt
Decode
1 : Enter Message, Select Image
2 : Enter Password
3 : Take Message,pwd & Image
4 : Embed Text & Image
5 : Add password
6 : Generate Stego
7 : Stego Sent to Reciever through some communication channel
8 : Browse Stego & Enter password
9 : Take Message & Pwd
10 : Tally with Original pwd
11 : If No Display wrong PWD
12 : If YES DECODE
13 : Display the Hidden Text

Fig3.11: sequence diagram for steganography

3.3.4. COLLABORATION DIAGRAM

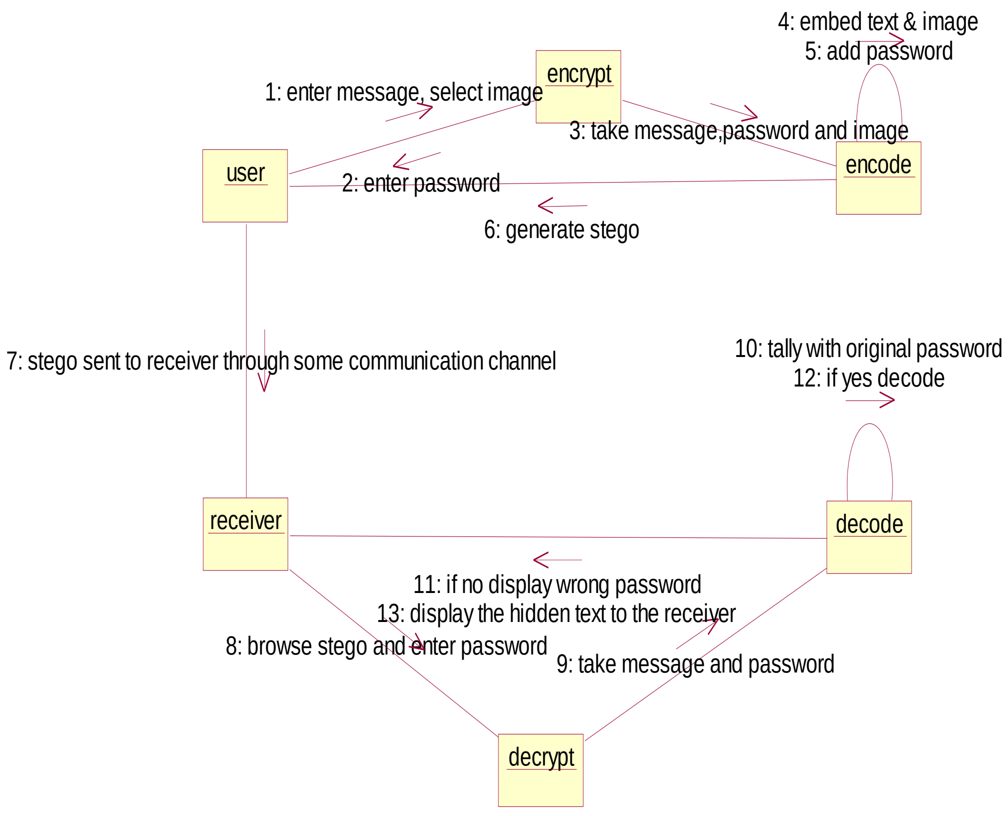

4: embed text & image
5: add password
encrypt
1: enter message, select image
3: take message,password and image
user
encode
2: enter password
6: generate stego
7: stego sent to receiver through some communication channel
10: tally with original password
12: if yes decode
receiver
decode
11: if no display wrong password
13: display the hidden text to the receiver
8: browse stego and enter password
9: take message and password
decrypt

Fig3.12: Collaboration diagram for steganography

3.3.5. ACTIVITY DIAGRAM

Activity diagram is essentially a flow chart, showing the flow of control from activity to activity. An activity is an ongoing non-atomic execution within a state machine. Activities ultimately result in some action, which is made up of executable atomic computations that result in a change in state of a system or the return of a value.

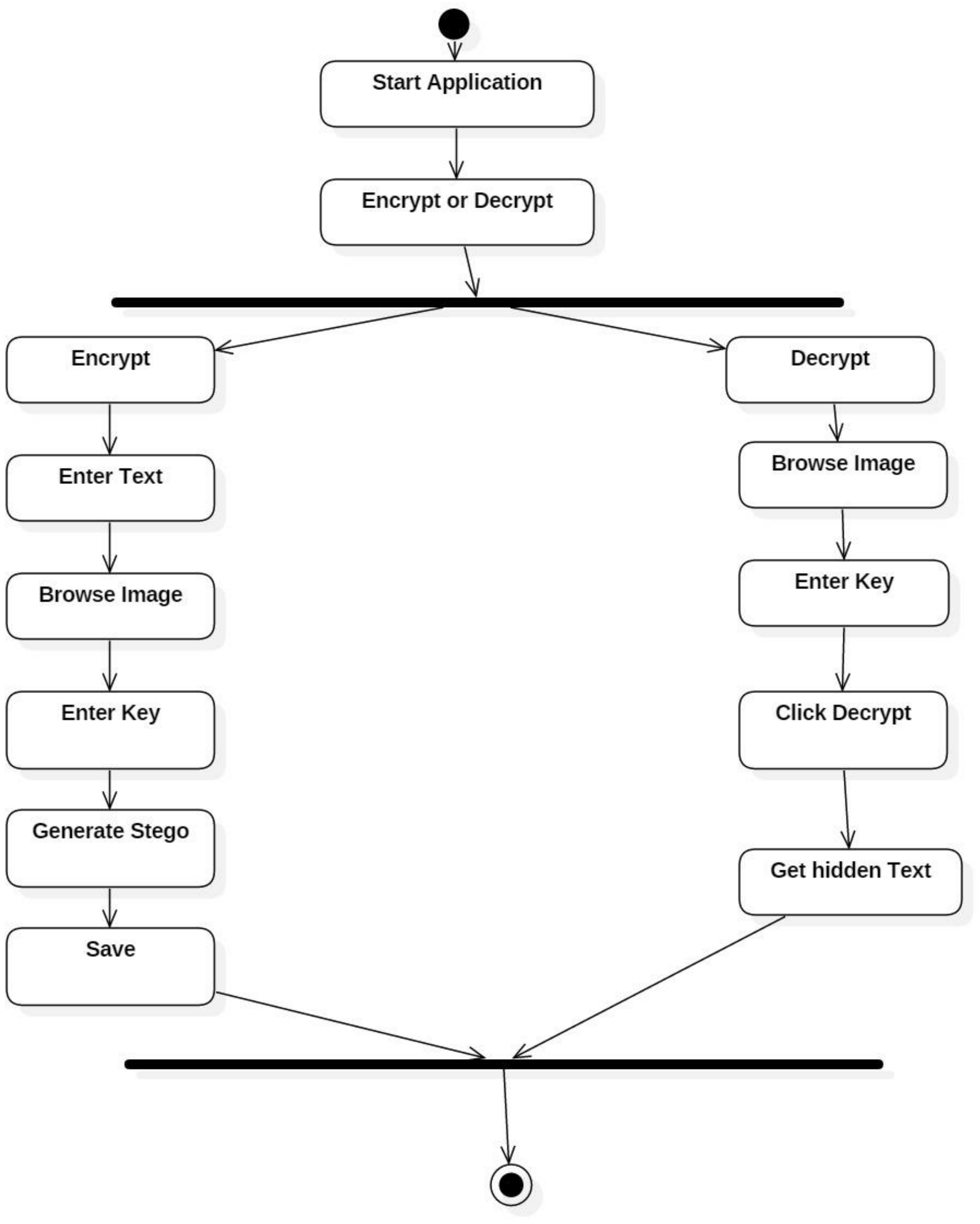


Fig3.13: Activity diagram for steganography

3.3.6. COMPONENT DIAGRAM

In the Unified Modeling Language, a **component diagram** depicts how components are wired together to form larger components and or software systems. The component diagram's main purpose is to show the structural relationships between the components of a system.

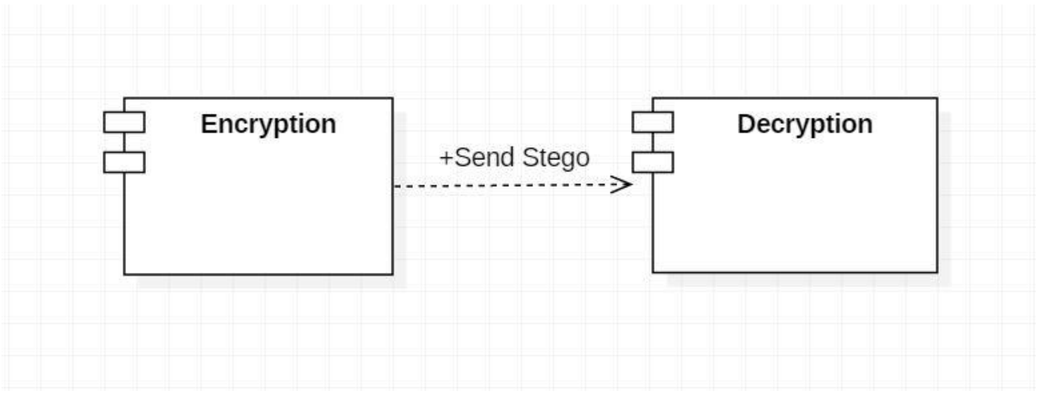


Fig3.14: Component diagram for steganography

4. TESTING

4.1 OVERVIEW OF TESTING

Software testing is a critical element of software quality and assurance and represents ultimate review of specifications, design and coding. Testing is an exposure of the system to trial input to see whether it produces correct output.

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also. The aim of testing is often to demonstrate that a program works by showing that it has no errors. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn't work.

4.1.1 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 the 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 a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards

4.2 TEST CASE DESIGN

The design of tests for software and other engineered products can be as challenging as the initial design of the product itself. Yet, software engineers often treat testing as an afterthought, developing test cases that may "feel right" but have little assurance of being complete. Recalling the objectives of testing, we must design tests that have the highest likelihood of finding the most errors with a minimum amount of time and effort.

A rich variety of test case design methods have evolved for software. These methods provide the developer with a systematic approach to testing. More important, methods provide a mechanism that can help to ensure the completeness of tests and provide the highest likelihood for uncovering errors in software.

4.2.1. TESTING STRATEGY

4.2.1.1. Code Testing

This strategy examines the logic of the program. To follow this method we developed some test data that resulted in executing every instruction in the program and module i.e. every path is tested. Systems are not designed as entire nor are they tested as single systems. To ensure that the coding is perfect two types of testing is performed or for that matter is performed or that matter is performed or for that matter is performed on all systems.

Example:

Module name	Condition	Action
Add Text	Text Added	Displays the message as “Text Entered”
Add Image	Image added	Displays the message as “Image added”

Table 4.1 Code Testing

4.2.1.2. Link Testing

Link testing does not test software but rather the integration of each module in system. The primary concern is the compatibility of each module. The Programmer tests where modules are designed with different parameters, length, type etc.

Example:

Module1	Module2	Parameters	Action
Encrypt	Encode	Img,tt2,pwd…etc	Compatible
Decrypt	Decode	Img2,dpwd…etc	Compatible

Table 4.2 Link Testing

4.2.1.3. Black Box Testing

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

Example:

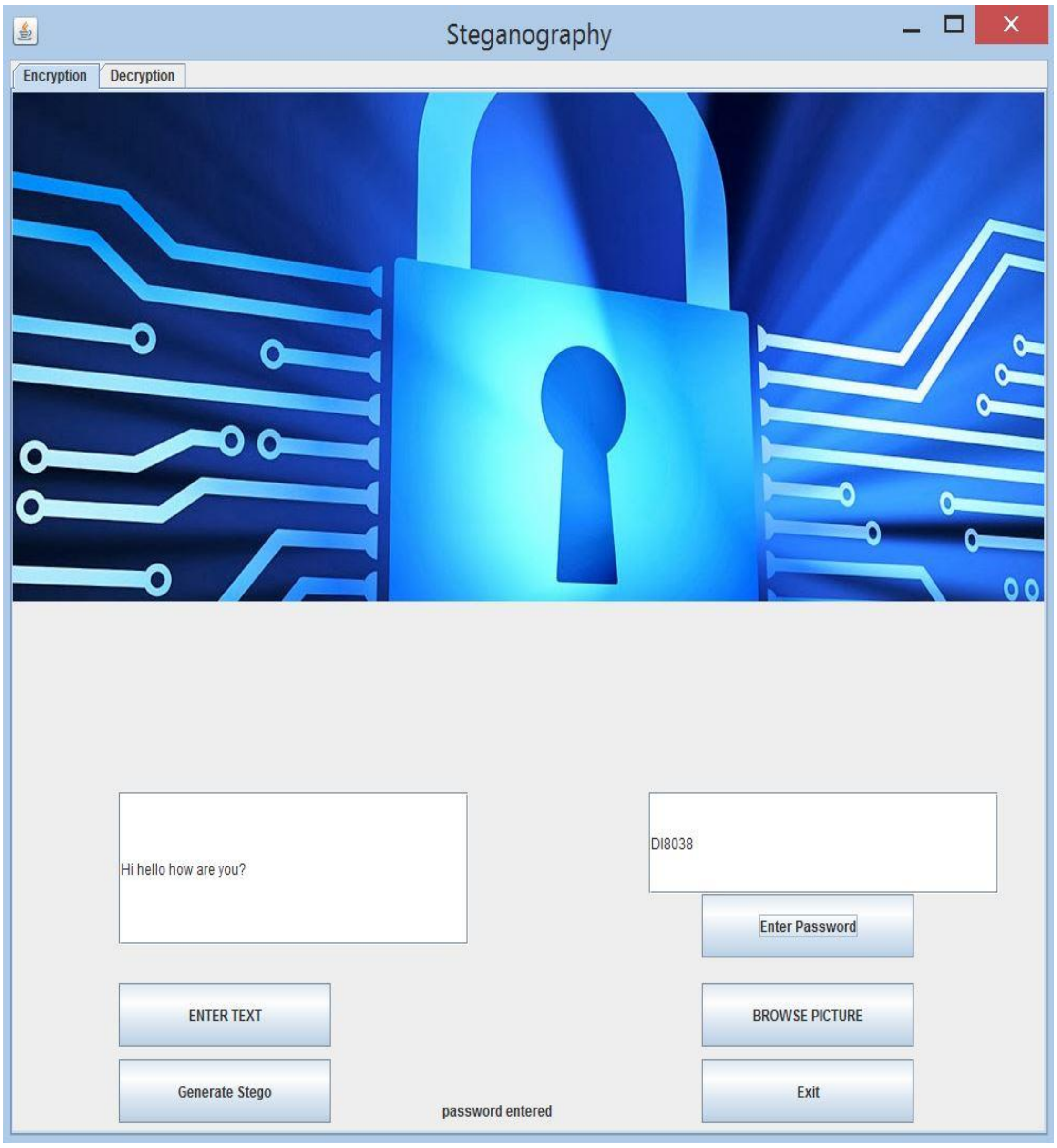
BlackBox1	Input	Output	BlackBox2	Action
Encrypt	Img,tt2,pwd …etc	Img,tt2,pwd …etc	Encode	Successfully forwarded

Table 4.3 Black Box Testing

4.2.1.4. Acceptance Testing

The final level of testing is the acceptance testing. Acceptance testing provides the users with assurance that the system is ready for production use; it is performed by the users. It uses the System Requirements document as its source. There is no formal documentation required for acceptance testing. Acceptance testing checks the system against the “Requirements”. It is similar to systems testing in that the whole system is checked but the important difference is the chance in focus.

5. SCREENSHOTS



Hi hello how are you?

DI8038

password entered

Fig 5.1: Enter Text and Password

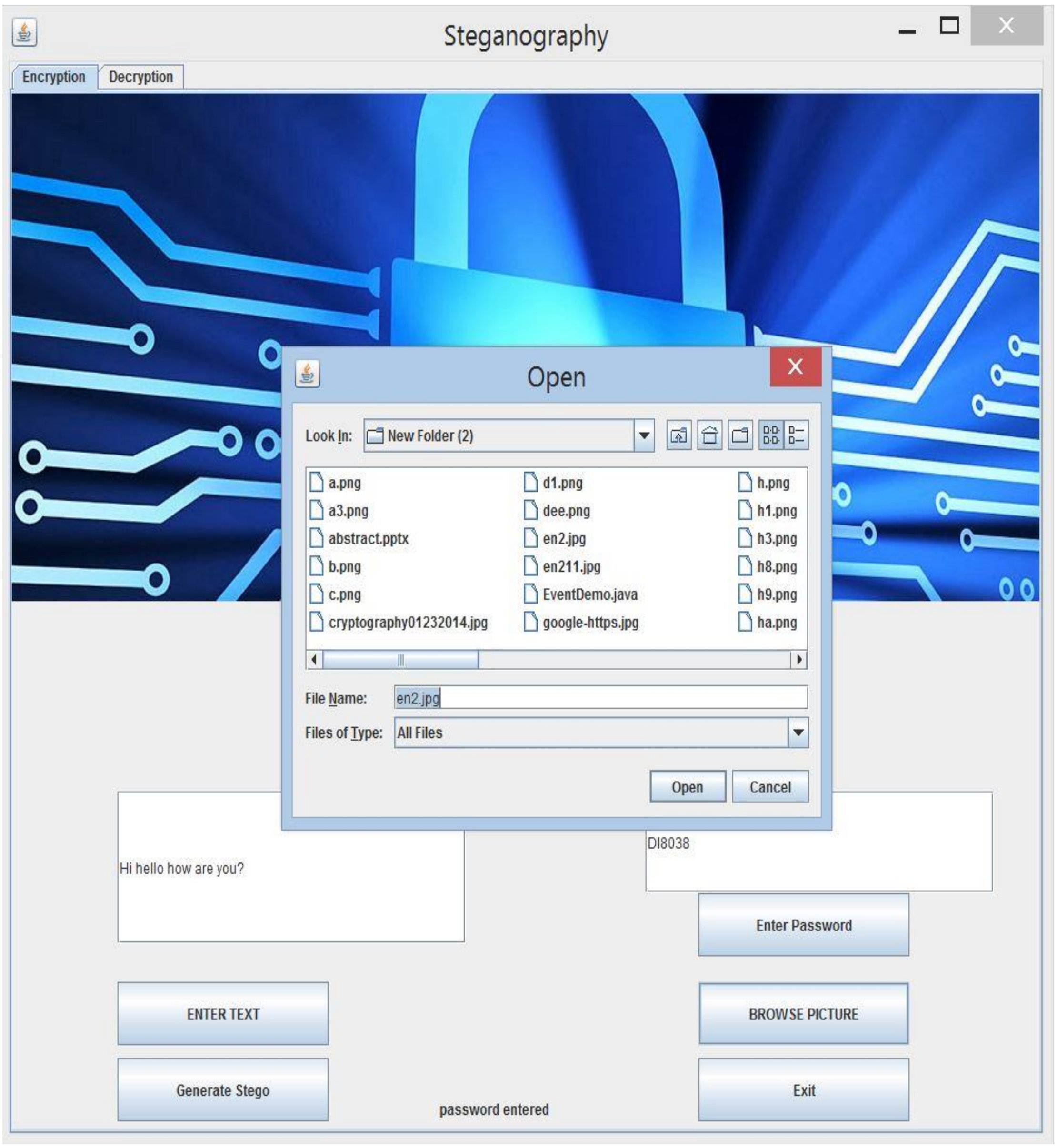


Fig 5.2: Select image file

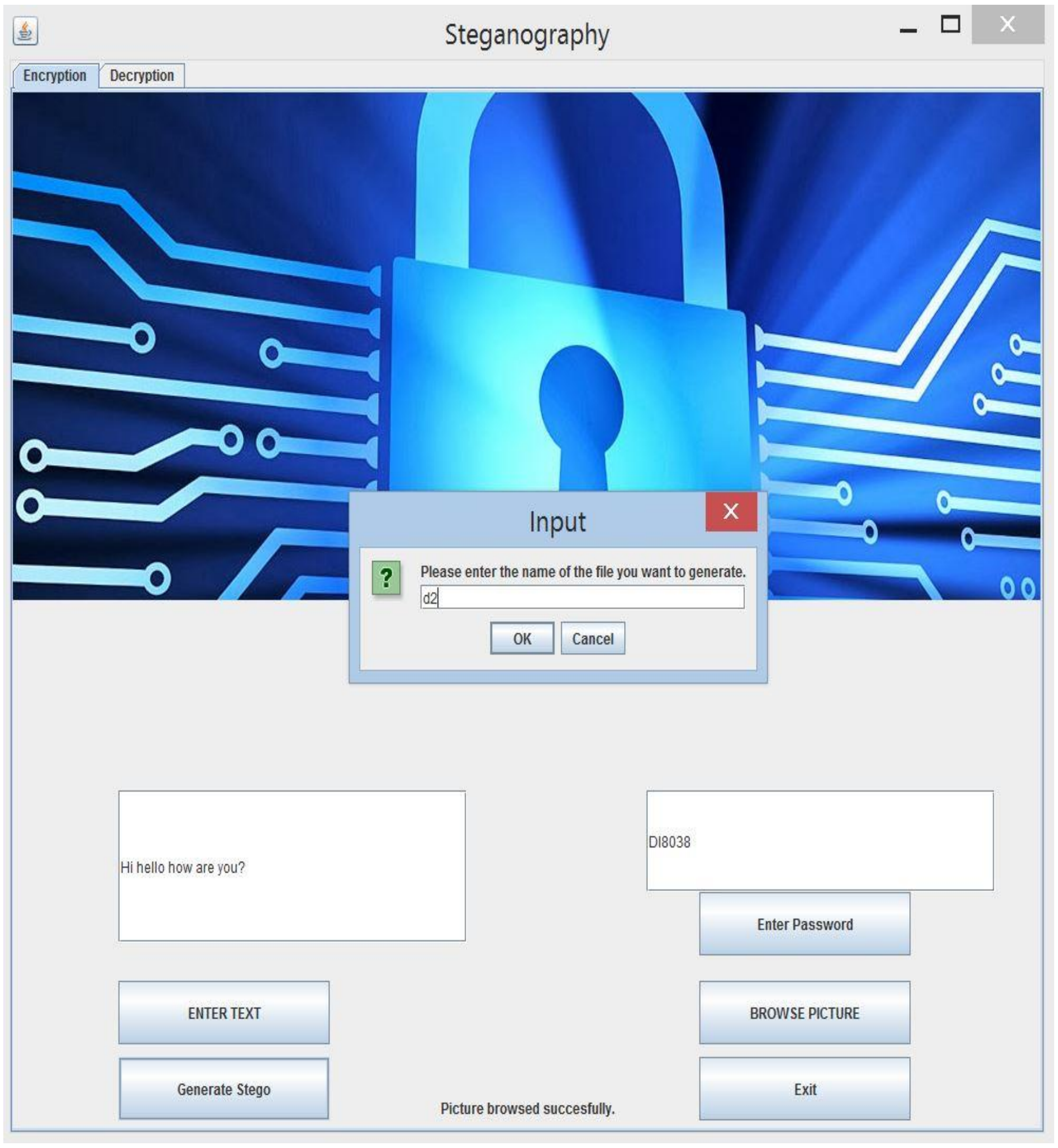
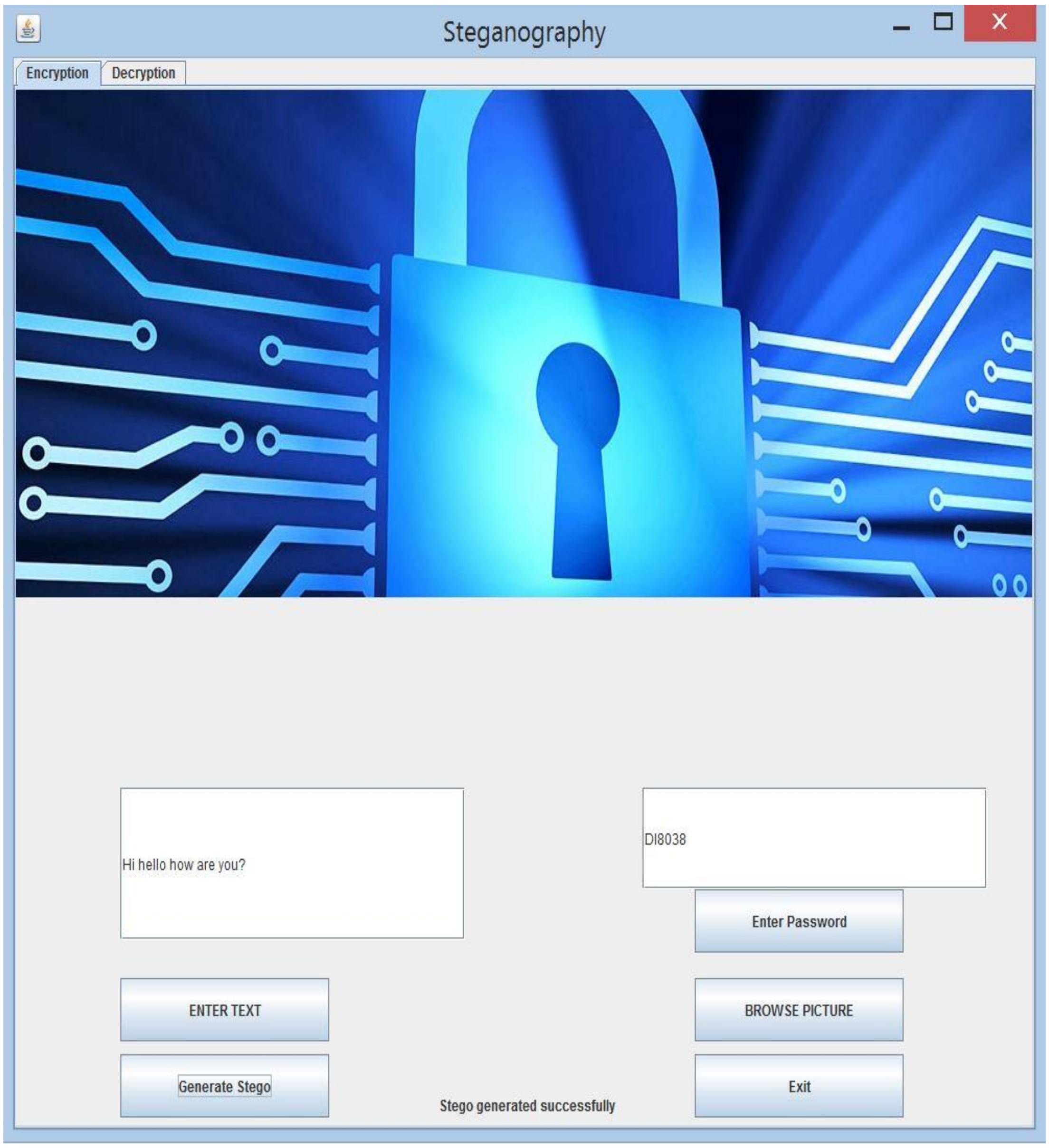


Fig 5.3: Save stego image file with new name



Hi hello how are you?

DI8038

Stego generated successfully

Fig 5.4: Generate stego image

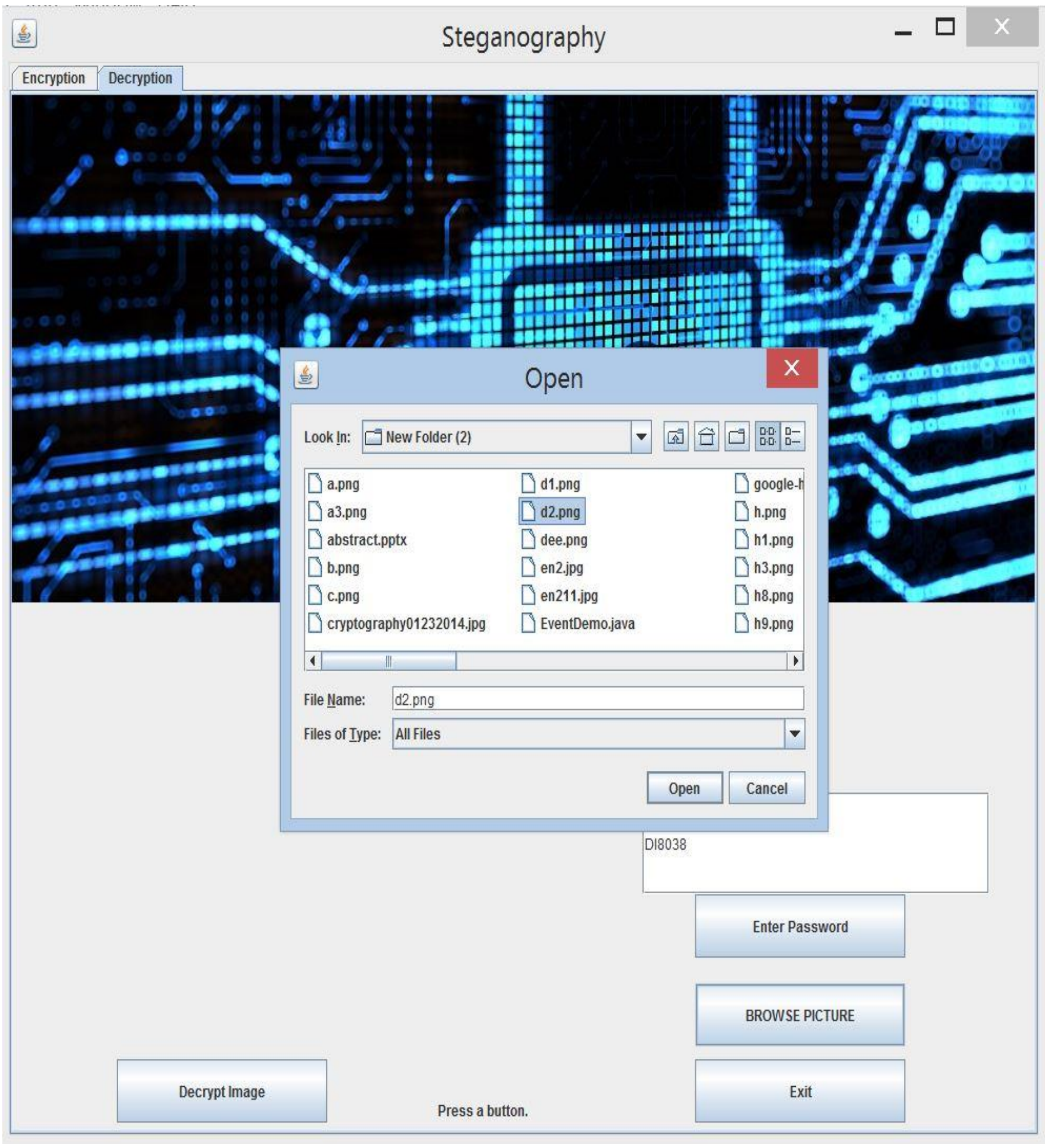
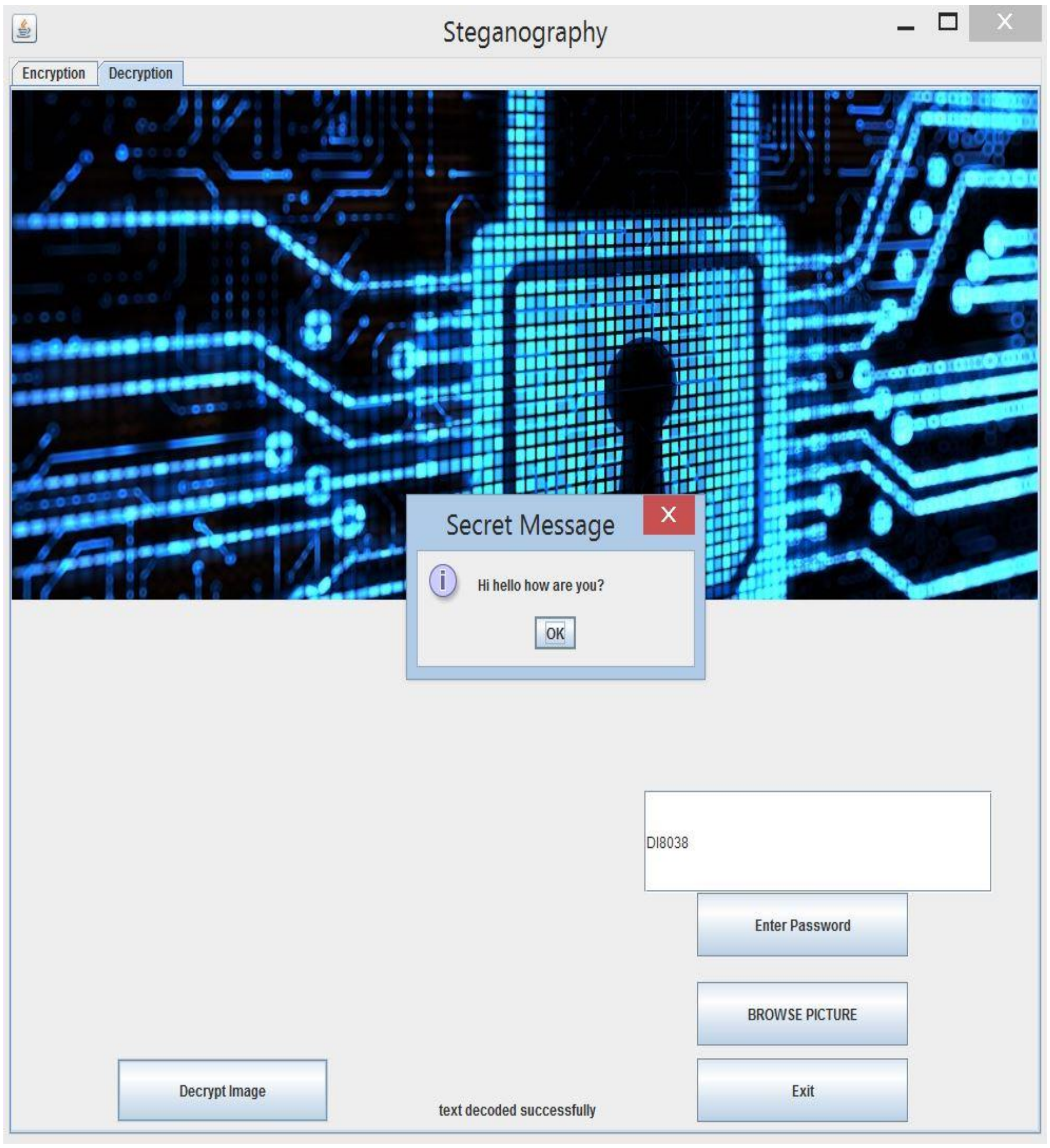


Fig 5.5: Select file to decrypt and enter password



Hi hello how are you?

DI8038

text decoded successfully

Fig 5.6: Decrypt Stego image

6. CONCLUSION

With the help of this procedure we developed in this steganography application we can successfully embed a secret confidential message into a picture along with added security measures of using a random translated user entered embedded password and also further decode and retrieve the hidden text embedded into the picture when the receiver enters a matching correct shared key. This process helps in use-age of this application successfully into defence and other strategic security organizations. The passage of information in such organizations with confidentiality in highly ever growing need and important. Thus this application helps in satisfying needs of such organizations and environments.

7. FUTURE ENHANCEMENTS

- This application can be extended by further improving the security by implementing more randomness in password embedding.
- This System can be implemented as a Mobile based application.

9. BIBILOGRAPHY & REFERENCES

- Fridrich, Jessica; M. Goljan; D. Soukal (2004). "Searching for the Stego Key"(PDF). Proc. SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents VI 5306: 70–82. Retrieved 23 January 2014.
- Pahati, OJ (2001-11-29). "Confounding Carnivore: How to Protect Your Online Privacy". AlterNet. Archived from the original on 2007-07-16. Retrieved 2008-09-02.