ISSN No : 2454-4221 (Print)
ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

SECURE TRANSMISSION OF DATA USING IMAGE STEGANOGRAPHY

M.S.S.Vahini [1], K. Kavya [2], J.Tejamani [3], M. Lakshmi Manaswini [4], T.Pavani [5].
1 Assistant Professor, Dept. of CSE, NS Raju Institute of Technology Visakhapatnam, Andhra Pradesh, India
2,3,4,5 Student, Dept of CSE, NS Raju Institute of Technology Visakhapatnam, Andhra Pradesh, India
vahini.cse@gmail.com, kavyarao175@gmail.com, jamitejamani99@gmail.com, manaswinimuggu@gmail.com, pavanithanavarapu49@gmail.com.

Abstract
Steganography is the art of hiding the fact that communication is taking place, by hiding information in other information. Many different carrier file formats can be used, but digital images are the most popular because of their frequency on the Internet. For hiding secret information in images, there exists a large variety of steganography techniques some are more complex than others and all of them have respective strong and weak points. Different applications have different requirements of the steganography technique used. While cryptography provides privacy, steganography is intended to provide secrecy. The aim of steganography is to hide the secret messages and also for communication and transferring of data. Steganography is also used in transferring the information of credit card or debit card to e-commerce for purchasing items. So no one apart from the authorized sender and receiver will be aware of the existence of the secret data.

Keywords: Mobile App, Common Home Utility, Wi-fi is ubiquity.

1. Introduction

In the physical world when we want to hide an object, we wrap it in some unmarked paper. This work is similarly done in the digital world but here we have the pixels in the cover image instead of wrap paper. Data hiding is referred to as a process to hide data (representing some information) into cover media. That is, the data hiding process links two sets of data, a set of the embedded data and another set of the cover media data. The relationship between these two sets of data characterizes different applications [1]. Because of the human visual system's low sensitivity to small changes and the high flexibility of digital media, one can easily make small changes in digital data with low perceptibility. Consequently, many interesting applications of data hiding in digital media can be created, like watermarking, caption data embedding, secret message delivery, etc. The purposes and requirements of such applications vary and are illustrated in Table 1 [2] [3] [4]. Drawing analogy between data hiding and communication, the embedding methods serve as a physical communication layer, on top of which other functionalities and features can be built.

Cite this article as: M.S.S.Vahini, K.Kavya, J.Tejamani, M.Lakshmi Manaswini & T.Pavani., "Secure Transmission Of Data Using Image Steganography", International Journal of Research in Advanced Computer Science Engineering, (IJRACSE), Volume 6 Issue 11, April 2021, Page 1-13.

For instance, security issues may be handled by top layers in authentication applications. In these models, the major objective of an adversary is to forge authentication data so that an altered document can still pass authentication tests [5]. Most of these algorithms work in frequency-domain [6]. However working in frequency- domain is very complex and takes more time and makes more changes in the cover image. In this paper we introduce a new algorithm that works in spatial-domain instead of frequency domain. Finding same pixels in the secret image and the cover image is a difficult problem. Our method solves this problem by employing the difference of adjacent pixels as the main metric. We propose a new method to embed a grey-valued image into a grey-valued cover image. It can be used for delivering secret images such as confidential images, military maps, etc. The method is designed in such a way that the grey value of every pixel of the secret image is preserved, i.e. no distortion will be created in the secret image when it is extracted out from the cover-image. On the other hand, because the resulting cover-image usually contains a large quantity of embedded data, it may 2 seriously be degraded. The proposed method, however, produces indistinguishable changes in the resulting cover-image. Since the precision of pixel grey values may be destroyed when transforming them back and forth between the frequency and spatial domains, we adopt a way of embedding the secret image into the cover image directly in the spatial domain.

1.1 Methodology

Basic of method Now, perform the operation of ‘image difference‘ to the two images in the following way: for every pixel value gi in either image, where i = 1, 2,. . . , let gi'=gi - gi-1 + 128 where gi-1, is the grey value of a neighboring pixel to gi , and create a new image with all gi' as the new pixel values and call it the difference image. Adding 128 is for normalizing the value and reducing the number of negative value. The details of image differencing (ID) in our proposed method, which are a little different from what just mentioned, will be in the following section. The pixel values of the difference images are concentrated near 128 resulting from the grey-value similarity between adjacent pixels. The histograms of the difference images are quite similar in shape, in contrast with those of their original images, which are quite different. This similarity helps us exploit a way to embed a secret image easily by grey value replacement using the difference metric. The amount of pixel-value change in a smooth image area is smaller than that in an edge or high-contrast area. Edge or high-contrast areas in general show greater changes. We use this characteristic to hide more data in high-contrast areas and less in smooth areas. This helps to hide more data indistinguishable in an image [9] [10]. The proposed hiding method consists of three major parts: 1) cover ID processing 2) secret ID processing and 3) the embedding process, which are described next.

1.2 ID For Cover and Secret Images

Cover and secret images used in the proposed method are assumed to have 256 grey values. To get a difference image from a given cover image, we obtain the grey value G of a pixel in the resulting image from every non-overlapping two-pixel of the cover image, in a zigzag order, di = [gi+1-gi+128] mod 256 (1) gi+1 and gi are the grey values of pixels, respectively in zigzag order. Adding 128 is for normalizing the value and reduce the number of negative value, and MOD 256 is for normalizing the values to the range of [0 to 255]. This quality way the number of out of range pixels become very small and this normalization doesn't destroy the images. The resulting data stream is just a half of the cover image in size.

The process of obtaining the difference image from the secret image is the same as that for the cover image except that we calculate the difference value of the grey values of every two adjacent pixels, instead of every non-overlapping two-pixel, in zigzag order. So the size of the resulting stream is the same as that of the secret image. We save the gray value of top-leftmost pixel of the secret image. Other 4 differencing methods assume that this value is 128[6][11]. In the sequel we call the difference image from the secret image as the secret difference image, and that from the cover image as the cover difference image. We create smaller ranges near 128 and larger ones far from 128 for the purpose of better replacement with less noticeable results. This technique helps us to embed more information in high-contrast areas with less deterioration of the image.

2. The Study of the System:

2.1 System Analysis

To conduct studies and analyses of an operational and technological nature, and To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. In the context of systems design are included i.e., Entity Relationship Diagrams.

Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement.
2. Output requirements.
3. Storage requirements.
4. Processing Requirements.
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:
1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with how the system presents information back to them. To use an analogy, a personal computer's physical design 10

a. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring

that each output element is designed so that people will find the system can use easily and effectively.
b. Select methods for presenting information.
c. Create document, report, or other formats that contain information produced by the system.

Involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc.

2.2 Input & Output Representation:

Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.
Input Design considered the following things:
What data should be given as input?
How the data should be arranged or coded?
The dialog to guide the operating personnel in providing input. Methods for preparing input validations and steps to follow when error occur.

Objectives

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. Efficient and intelligent output design improves the system's relationship to help user decision-making. The output form of an information system should accomplish one or more of the following objectives.

2.3 Requirements Definition Stage and Analysis:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

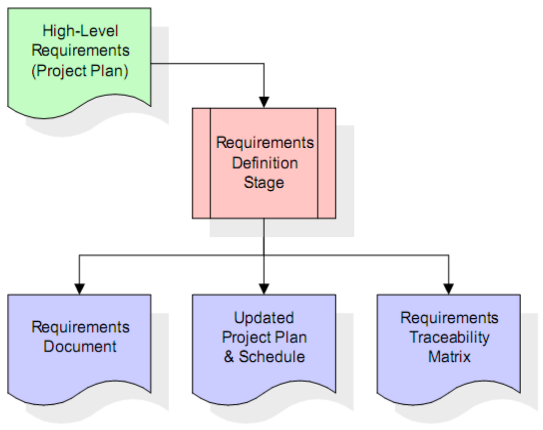

High-Level Requirements (Project Plan)
Requirements Definition Stage
Requirements Document
Updated Project Plan & Schedule
Requirements Traceability Matrix

Fig 1: Requirement Stage

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM).

3. Feasibility Study

Feasibility study is conducted once the problem is clearly understood. This study tells about how this package is useful to the users and its advantages and disadvantages, and also it tells whether this package is cost effective are not. There are three types of feasibility study, they are

• Economic Feasibility.
• Technical Feasibility.
• Operational Feasibility.

3.1 Technical Feasibility:

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed designs of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc.

3.2 Operational Feasibility:

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project: Have the user been involved in the planning and development of the project? Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

3.2 Economic Feasibility:

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision-making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

4. System Requirements

4.1 Functional Requirements:

Functional requirements describe what the system should do, i.e., the services provided for the users and for other systems.

Data Storage

In this project we are using My SQL as Data base storage in order to store the details of images into data base and in order to retrieve the data at the time of accessing. For this we are using My Sql data base as source for storing and retrieving the files.

Modules

After a deep analysis of the system requirements the project has been developed by getting divided into the following modules. Implementation is the stage of the project when the theoretical design is turned out into a working system.

ISSN No : 2454-4221 (Print)
ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

4.2 Performance Requirements:

Performance is measured in terms of the output provided by the application. Requirement specifications play an important part in the analysis of system. It is very difficult to change the system once it has been designed and on the other hand designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.The requirement specification for any system can be broadly stated as given below:

- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the duties.

4.3 Hardware Requirements:

System	: Pentium IV 2.4 GHz.
Hard Disk	: 1TB.
Floppy Drive	: 1.44 Mb.
Monitor	: 14’ Colour Monitor.
Mouse	: Optical Mouse.
Ram	: 8 GB Ram.
Keyboard	: 101 Keyboards.

4.4. Software Requirements:

Operating System : windows 10
Language : Python programming language

4.5 Document Conventions:

The following are the conventions and the acronyms that are used in this document. Styles and Formatting:

- Aligned At : Left of the page
- Text Font : Times New Roman, 12
- Heading Font: Times New Roman (Bold), 16
- Sub-heading Font : Times New Roman (Bold), 12

5. System Design

5.1 System Design Introduction

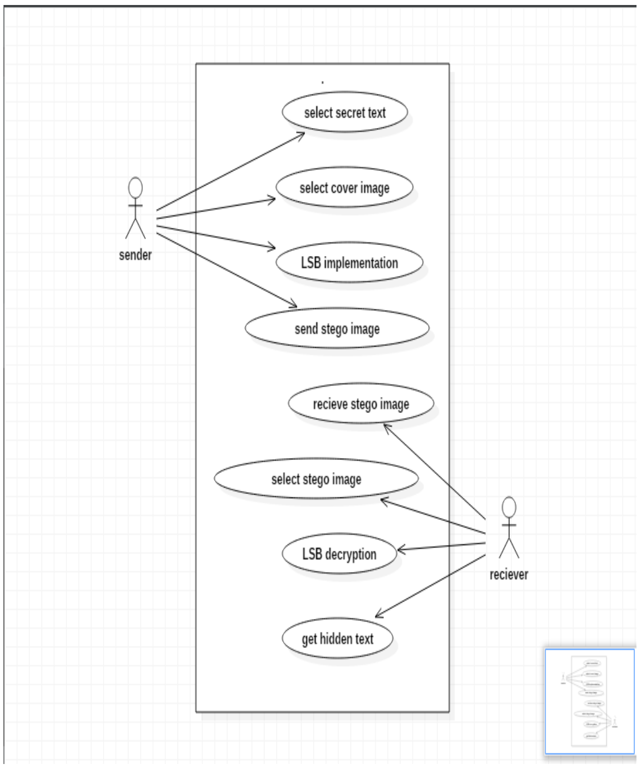
The Unified Modeling Language (UML) allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. The Unified Modeling Language is commonly used to visualize and construct systems which are software intensive. Because software has become much more complex in recent years, developers are finding it more challenging to build complex applications within short time periods. Even when they do, these software applications are often filled with bugs, and it can take programmers weeks to find and fix them. This is time that has been wasted, since an approach could have been used which would have reduced the number of bugs before the application was completed.

Since UML is not a methodology, it does not require any formal work products. Yet it does provide several types of diagrams that, when used within a given methodology, increase the ease of understanding an application under development. There is more to UML than these diagrams, but for my purposes here, the diagrams offer a good introduction to the language and the principles behind its use. By placing standard UML diagrams in your methodology's work products, you make it easier for UML-proficient people to join your project and quickly become productive.

Each view is defined by a set of diagrams, which is as follows.
User Model view
Structural Model View
Behavioral Model View
Implementation Model View
Environmental Model View

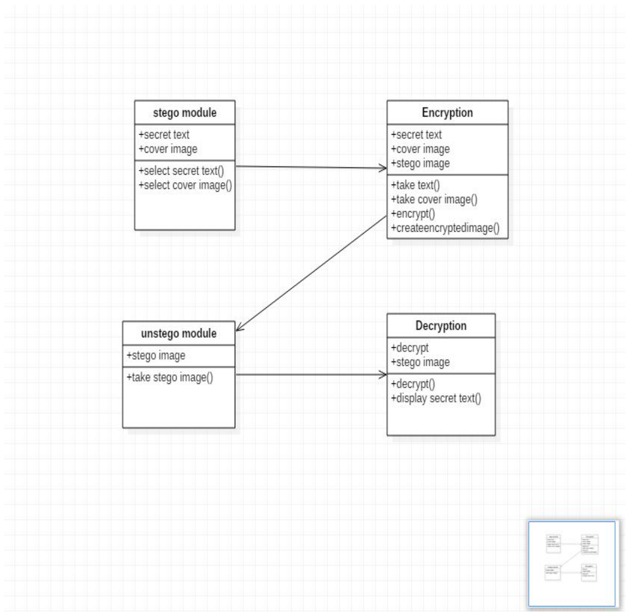
5.2 UML DIAGRAMS:

Use Case Diagram:



A Use Case Diagram at its simplest is a representation of a user's interaction with the system. First user writes secret text then he selects cover image and data gets hidden inside image, then user sends stego image to receiver through image. At the receiver side, user selects the stego image and applies decryption on stego image. After that he can get text hidden in the text.

CLASS DIAGRAM:



6. CODING

ENCRYPTION.py

```
import tkinter as tk
import PIL
from PIL import Image, ImageTk
from tkinter import Tk, file dialog, Label,
Button
from tkinter import *
from NumPy import *
from pylab import imread, imsave
from tqdm import tqdm
from matplotlib. pyplot import imshow
import matplotlib. pyplot as plt
from random import choice
# Initial module tkinter.Tk as main root
master = Tk ()
master["bg"] = "#4aa96c"
master.geometry("800x600")
master.title("Image Encrypt GUI")
key = []
# function
def enc(imgdir, key):
fin = open(imgdir, "rb")
img = fin.read()
```

```
fin.close()
image = bytearray(img)
for index, value in enumerate(image):
image[index] = key ^ value
fin = open (imgdir, "wb")
fin. write(image)
def addtext():
inp = input_key.get(1.0, "end-1c")
key.append(inp)
# main content
label = Label(master, text="Image Encrypt
Apps",font=("Arial", 36),fg="white",
bg="#9fe6a0", height=2, width=30)
label.place(x=0, y=0)
canva_frame = Frame(master, width=300,
height=250, bg="white").pack_propagate(0)
canva_img1 = Label(canva_frame,
bg="white")
canva_img1.place(x=85, y=150)
canva_img2 = Label(canva_frame,
bg="white")
button_input = Button(text="Add Key",
bg="white", command=addtext)
button_input.place(x=180, y=397)
button_filedir = Button(text="Browse Image",
bg="white", command=askfile)
button_filedir.place(x=85, y=440
master.mainloop()
```

GUI_DECRYPTE.py

```
import tkinter as tk
import PIL
from PIL import Image, ImageTk
from tkinter import Tk, file dialog, Label,
Button
from tkinter import *
from NumPy import *
from pylab import imread, imsave
from tqdm import tqdm
from matplotlib.pyplot import imshow
import matplotlib.pyplot as plt
from random import choice
# initial module tkinter.Tk as

main root
master = Tk()
master["bg"] = "#4aa96c"
master.geometry("800x600")
master.title("Image Decrypt GUI")
key = []
# function
def enc(imgdir, key):
fin = open(imgdir, "rb")
img = fin.read()
fin.close()
image = bytearray(img)
for index, value in enumerate(image):
image[index] = key ^ value
fin = open(imgdir, "wb")
fin.write(image)
fin.close()
def askfile():
# ask filename
)
canva_img1.configure(image=image_ori)
canva_img1.img = image_ori
# encrypt
for value in key:
print(value)
enc(path, int(value))
key.clear()
# show image result
img_res = PIL.Image.open(path)
img_res = img_res.resize((300,230))
image_ori_res                          =
ImageTk.PhotoImage(img_res)
canva_img2.configure(image=image_ori_res)
canva_img2.img_res = image_ori_res
```

```
def addtext():
inp = input_key.get(1.0, "end-1c")
key.append(inp)
# main content
label = Label(master, text="Image Decrypte Apps",font=("Arial", 36),fg="white", bg="#9fe6a0", height=2, width=30)

label.place(x=0, y=0)
canva_frame = Frame(master, width=300, height=250, bg="white").pack_propagate(0)
canva_img1 = Label(canva_frame, bg="white")
canva_img1.place(x=85, y=150)
canva_img2 = Label(canva_frame, bg="white")
canva_img2.place(x=420, y=150)
input_key = Text(master, width=10, height=1)
input_key.place(x=85, y=400)
button_input = Button(text="Check Key", bg="white", command=addtext)
```

7. System Testing

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.

7.1.1 Levels of Testing:

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

7.1.2 CODE TESTING:

This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified.

7.1.3 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.

7.1.4 UNIT TESTING:

In the unit testing we test each module individually and integrate with the overall system. 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
2. White Box Testing

7.2 SYSTEM TESTING:

The top-down testing, which began from upper level to lower-level module, was carried out to check whether the entire system is performing satisfactorily. There, are three main kinds of System testing:
1 Alpha Testing
2 Beta Testing
3 Acceptance Testing

Alpha Testing: This refers to the system testing that is carried out by the test team with the Organization.
Beta Testing: This refers to the system testing that is performed by a selected group of friendly customers
Acceptance Testing: This refers to the system testing that is performed by the customer to determine whether or not to accept the delivery of the system.

ISSN No : 2454-4221 (Print)
ISSN No : 2454-423X (Online)
International Journal of Research in Advanced Computer Science Engineering
A Peer Reviewed Open Access International Journal
www.ijracse.com

7.3 Integration Testing:

Data can be lost across an interface, one module can have an adverse effort on the other sub functions, when combined, may not produce the desired major functions.

7.4Output Testing:

After performance of the validation testing, the next step is output testing. The output displayed or generated by the system under consideration is tested by asking the user about the format required by system.

7.5 Test Cases:

Test plan:

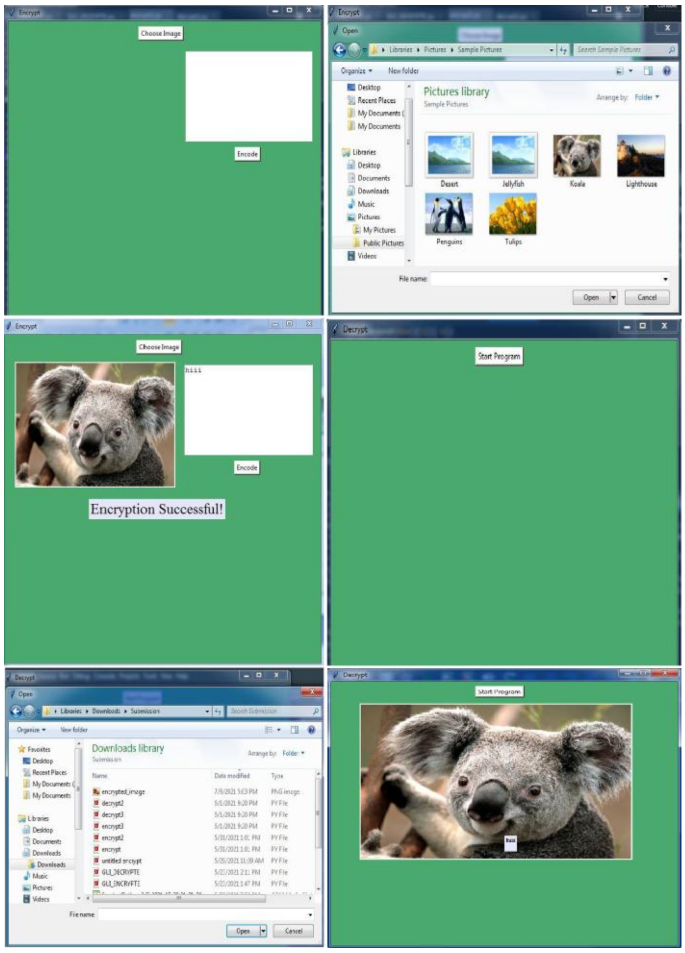
The test-plan is basically a list of test cases that need to be run on the system. Some of the test cases can be run independently for some components (report generation from the database, for example, can be tested independently) and some of the test cases require the whole system to be ready for their execution.

Test Case 1:

Table 1 Test case for Browsing input file

Test Aspect	Browsing for Input File
Test Data	JPEG
Expected Result	Image Uploaded Successful
Condition being checked	Checking for valid extensions of images
Action Taken	If not, Upload Process will be not possible

8. SCREENS



9. Maintenance

Maintenance the last phase in the software engineering process. They are

1 Adaptive Maintenance
2 Corrective Maintenance
3 Perfective Maintenance
4 Preventive Maintenance

Adaptive Maintenance is applied when changes in the external environment precipitate modifications to software. It deals with adapting the software to new environments.

Perfective Maintenance incorporates enhancements that are requested by user community. It deals with updating the software according to changes in user requirements.

Corrective Maintenance acts to correct errors that are uncovered after the software is in use. It deals with fixing bugs in the code.

Preventive Maintenance improves future maintainability and reliability and provides a basis for future enhancement. It deals with updating documentation and making the software more maintainable.
There are four major problems that can slow down the maintenance process.
1. Unstructured Code
2. Maintenance programmers having insufficient knowledge of the system
3. Documentation being absent
4. Out of Date, or at best insufficient

Performance
Our system which is developed in Jsp and Servlet technology is a versatile product and is platform independent. The features provided by the Online System makes it one of an interactive online platform for face annotation.

Future Scope
Steganography though is still a fairly new idea. There are constant advancements in the computer, suggesting advancements in the steganography as well. It is likely that there will soon be more client and more advanced techniques for Steganalysis. A hopeful advancement is the improved sensitivity to small messages. Knowing how difficult it is to detect the presence of a fairly large text _le within an image, imagine how difficult it is to detect even one or two sentences embedded in an image! It is like a microscopic needle in the ultimate haystack. What is scary is that such a small _le of only one or two sentences may be all that is needed to commence a terrorist attack. In the future, it is hoped that the technique of Steganalysis will advance such that it will become much easier to detect even small messages within an image. In this work it explores only a small part of the science of steganography. As a new displine, there is a great deal more research and development to do. The following section describe areas for research which were shoots of, or tangential to, our main objectives.

1. Detecting Steganography in Image Files:
Can steganography be detected in images _les? This is difficult question. It may be possible to detect a simple Steganographic technique by simple analyzing the low order bits of the image bytes. If the Steganographic algorithm is more complex, however, and spreads the embedded data over the image is random way or encrypts the data before embedding, it may be nearly impossible to detect.

2. Steganography on the World Wide Web:
The world wide web(www) makes extensive use of inline images. There are literally millions of images on various web pages worldwide. It may be possible to develop an application to serve as a web browser to retrieve data embedded in web page images. This stego-web could operate on top of the existing WWW and be a means of covertly disseminating information.

3. Steganography in printed media:

If the data is embedded in an image, the image printed, then scanned and stored in a _le can the embedded data be recovered? This would require a special form of a steganography to which could allow for in accuracies in the printing and scanning equipment.

10. Conclusion

A simple-to-implement yet effective method has been proposed in this paper for image encryption using a combination of different permutation techniques. The main idea stems from the fact that the perceivable information in an image can be reduced by decreasing the correlation among the bits, pixels and blocks using certain permutation techniques. This paper has presented an approach for a random combination of the aforementioned permutations for image encryption. From the results, it is observed that combined method achieves the advantages all individual permutation techniques and overcomes the limitations of these methods, e.g., visual intelligence and redundancy. A further extension of the approach is possible by using variable length keys with the information available to the receiver via PRIG indices, thereby increasing the level of security significantly.

References

[1] A. J. Elbert and C. Paar, "An Instruction-Level Distributed Processor for Symmetric-Key Cryptography," IEEE Trans. Parallel and distributed systems, vol. 16, no. 5, pp. 468-480, May 2005.

[2] W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Trans. Information Theory, vol. 22, no. 6, pp. 644-654, Nov. 1976.

[3] W. Stallings, Cryptography and Network Security. Englewood Cliffs, NJ: Prentice Hall, 2003.

[4] E. Besdok, "Hiding information in multispectral spatial images," Int. J. Electron. Common. (AEU) 59, pp. 15-24, 2005.

[5] S. Trivedi and R. Chandramouli, "Secret Key Estimation in Sequential Steganography," IEEE Trans. Signal Processing, vol. 53, no. 2, pp. 746- 757, Feb. 2005.

[6] Y. Wu, "On the Security of an SVD-Based Ownership Watermarking," IEEE Trans. Multimedia, vol. 7, no. 4, pp. 624-627, Aug. 2005.

[7] Y. T. Wu and F. Y. Shih, "An adjusted-purpose digital watermarking technique," Pattern Recognition 37, pp. 2349-2359, 2004.

[8] A. Masoud and A. H. Tewfik, "Geometric Invariance in Image Watermarking," IEEE Trans. Image Processing, vol. 13, no. 2, pp. 145-153, Feb. 2004.

[9] S. S. Maniccam and N. G. Bourbakis,"Image and video encryption using scan patterns," Pattern Recognition 37, pp. 725-737, 2004.

[10] P. P. Dang and P. M. Chau, "Image Encryption for Secure Internet Multimedia Applications," IEEE Trans. Consumer Electronics, vol. 46, no. 3, pp. 395-403, Aug. 2000.

[11] W. Zeng and S. Lei, "Efficient Frequency Domain Selective Scrambling of Digital Video," IEEE Trans. Multimedia, vol. 5, no. 1, pp. 118-129, March 2003.

[12] L. T. Wang and E. J. McCluskey, "Linear Feedback Shift Register Design Using Cyclic Codes," IEEE Trans. Computers, vol. 37, no. 10, pp. 1302- 1306, Oct. 1988.

[13] A. Fuster and L. J. Garcia, "An efficient algorithm to generate binary sequences for cryptographic purposes," Theoretical Computer Science 259, pp. 679-688, 2001.