SECURING ORGANIZATIONAL DATA FROM MALICIOUS CODE

By

Madhav Dhingra	**(081259)**
Jai Sisodia	**(081251)**
Nikhita Jain	**(081257)**
Rahul Sharma	**(081448)**

Under the Supervision of

Mr Pradeep Kumar Gupta

Submitted in partial fulfilment

Of the requirements for the degree of

BACHELOR OF TECHNOLOGY (CSE)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT

SOLAN, HIMACHAL PRADESH INDIA

2012

विद्या तत्व ज्योतिसम:

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT

SOLAN, HIMACHAL PRADESH

CERTIFICATE

This is to certify that the work entitled **"SECURING ORGANIZATIONAL DATA FROM MALICIOUS CODES"**, submitted by **Madhav Dhingra, Jai Sisodia, Nikhita Jain** and **Rahul Sharma** in partial fulfilment for award of degree of Bachelor of Technology (Computer Science & Engineering) of Jaypee University of Information Technology has been carried out in my supervision. This work has not been submitted partially or wholly to any other university or institution for award of this or any other degree program.

[illegible]

Project Supervisor

Mr Pradeep Kumar Gupta

Sr. Lecturer (CSE & IT), JUIT

Date: 29/05/12

ACKNOWLEDGEMENT

We are highly grateful to **Brig. (Retd.) S.P. Ghrera**, Associate Professor and Head, Dept. of Computer Science & Engineering and IT, for providing us the opportunity to work on the project.

With great pleasure we express our gratefulness to our guide and mentor **Mr P.K.Gupta**, Senior Lecturer, Dept. of Computer Science & Engineering and IT, for his valuable and sustained guidance and careful supervision during the project.

We express our sincere thanks to all the faculty members of JAYPEE UNIVERSITY OF INFORMATION AND TECHNOLOGY who have helped us directly or in directly. Without their help and guidance we would not have been able to successfully complete our project.

Group Members:

Madhav Dhingra Madhav Dhingra
Jai Sisodia [illegible]
Nikhita Jain Nikhita
Rahul Sharma [illegible]

Abstract

The project undertaken by us introduces a method to secure the digital information as a watermark by using the characteristics of DNA for encryption of data. The watermarked image is embedded into the image as binary information and further encrypted as DNA sequences and these DNA sequences (after being grouped as tri - nucleotide sequences called codons) are then attached to particular proteins or amino acids to enhance redundancy of our data. Similarly during decryption the DNA coded data is converted into binary by using base conversion and this binary information is then used to reform the hidden data.

The concept behind this being simple that with the advent of new networking technologies there is increase in the number of harmful agents who want to access our data. Anti-Viruses, Anti-Rootkits and other Network Security Agents can't cope with the number of Malicious Codes & Software (Malware) being released every day. Thus, to secure our data, we introduce a method to encode data using an algorithm which converts binary data into DNA sequences. The key to decode the data is stored at receivers end. DNA sequences are attached to a particular protein according to a generated table, which enhances the redundancy of our data. This would result in higher level of security

Table of Contents

Chapter 1: Introduction

1.1. Introduction 01

1.2. Malicious Codes 02

1.2.1. Transmission and Infection 02

1.2.2. Categorization of Malware 03

1.2.3. Anti- Malware Programs 04

1.3. DeoxyRiboNucleic Acid 05

1.3.1. Structure of DNA 05

1.3.2. Proteins and Amino Acids 07

1.4. Discrete Wavelet Transform 09

1.5. Haar Wavelet 10

1.6. Aims and Objectives 11

1.7. Resource Requirements 12

1.7.1. Software Requirements 12

1.7.2. Hardware Requirements 12

1.8. Scope 12

1.9. Limitations 13

Chapter 2: Project Planning

2.1. UML Diagrams 14

2.1.1. Use Case Diagram 14

2.1.2. State Diagram 17

2.1.3. Sequence Diagram 19

2.2. Plan 20

2.3. Gantt Chart 21

Chapter 3: Algorithm Design

3.1. Watermark Embedding 22

3.2. Watermark Extraction 25

Chapter 4: Testing

4.1. The Goal of Testing 28

4.2. Significance Measures 28

4.2.1. Peak Signal to Noise Ratio (PSNR) 28

4.2.2. Structural Similarity Index Measure (SSIM) 29

4.3. The Testing Chart 30

4.4. For 4px Encoded Signature 32

4.5. For 8px Encoded Signature 34

4.6. For 16px Encoded Signature 36

Chapter 5: Conclusion

5.1. Results and Conclusion 38

5.2. Contribution of the project 38

References 39

APPENDIX: CODE 41

List of Figures

Fig. 1.1: Structure of DNA 6

Fig. 1.2: Relationship between DNA and Encoded Petide 7

Fig. 1.3: Subbandsof 2-D orthogonal wavelet transform 8

Fig. 1.4: Haar Wavelet 10

Fig. 3.1: Use Case Diagram 22

Fig. 3.2: State Diagram for Image Conversion to Array 23

Fig. 3.3: State Diagram of image for embedding signature onto it 23

Fig. 3.4: State diagram show steps of conversion of array to encoded array 24

Fig. 3.5: Sequence Diagram 25

Fig. 3.6: Gantt Chart 27

Fig. 4.1: Testing Information Flow 29

Fig. 4.2: Encoded Signature (4px) 30

Fig. 4.3: Lena (Watermarked, 4px Signature) 30

Fig. 4.4: Boat (Watermarked, 4px Signature) 31

Fig. 4.5: Cameraman (Watermarked, 4px Signature) 31

Fig. 4.6: Encoded Signature (8px) 32

Fig. 4.7: Lena (Watermarked, 8px Signature) 32

Fig. 4.8: Boat (Watermarked, 8px Signature) 33

Fig. 4.9: Cameraman (Watermarked, 8px Signature) 33

Fig. 4.10: Encoded Signature (16px) 34

Fig. 4.11: Lena (Watermarked, 16px Signature) 34

Fig. 4.12: Boat (Watermarked, 16px Signature) 35

Fig. 4.13: Cameraman (Watermarked, 16px Signature) 35

List of Tables

Table 3.1: Plan for Project 25
Table 4.1: Effect of varying the size of watermark 30

Chapter 1

Introduction

1.1. Introduction

Network administrators, Security personnel, and executive officers are finding it increasingly difficult to keep their organizational networks free from the debilitating and costly effects of malicious code. Because it can quickly spread throughout a network, understanding how to adequately protect these networks from the threat of malicious code is mandatory.

Results from Symantec published in 2008 suggested that "the release rate of malicious code and other unwanted programs exceed the release of legitimate software applications."[20] According to F-Secure, "In the previous 20 years as much malware was produced as in 2007"[21] Malware's most common mode of transmission from criminals to users is through the Internet. It is also the user's fault as they get attracted to false information.

Traditional encryption makes sure that even if data is access by unauthorized or malefic sources, the data appears to be garbled or random codes. This increases the probability that the data will come under the attacker's scrutiny, the attacker suspecting the data to be important. As more and more users can access the remote facilities and send or receive various kinds of digital data over the internet. The internet is public but insecure channel to transmit the data. Thus, to secure our data we introduce a novel method of encrypting our data using the concepts of DNA.

Nowadays various techniques deriving their inspiration from biology have become more and more popular, and they are being applied to many kinds of applications, authentication protocols [11], biochemistry and cryptography and so on [6, 11, 12]. The growth and emergence of new imaging technologies have allowed for techniques that can be used for copyright protection of digital data such as images, documents etc. [22].

1.2. Malicious Code

A Malicious Code any kind of software or code specifically designed to exploit a computer, or the data it contains, without consent. Malware or malicious software is software designed to disrupt computer operations, gather private information, or access a computer system without authorization. It can be in the form of script, code or even a software with a GUI[23]

It is the intent of creator of software which categorizes software as malware. It includes computer viruses, worms, Trojan horses, spyware, dishonest adware, scareware, crimeware, most rootkits, and other malicious and unwanted software or program.

The first internet worm and many other early infectious programs were written as experiments or pranks. Their purpose was not to harm the computer or the data stored but act as pranks or jokes. As engineers for more and more ways to exploit these existing "Loopholes", the ill intent became more prevalent. Examples can be found in programs designed to cause harm or data loss. Many DOS viruses, and the Windows ExploreZip worm, were designed to destroy files on a hard disk, or to corrupt the file system by writing invalid data to them. Network-borne worms such as the 2001 Code Red worm or the Ramen worm fall into the same category.

1.2.1. Transmission and Infection

Before Internet access became a widespread phenomenon, viruses spread on personal computers by infecting the executable boot sectors of floppy disks. Inserting a copy of it into the machine code instructions in these executables a virus itself,when we run the program or when the disk is booted.The first generation of viruses were written targeting the Apple II and Macintosh operating system, but with the exponential increase in the dominance of the IBM PC and MS-DOS system virus became more focussed towards these operating systems. Executable-infecting viruses are dependent on users who exchange software or boot-able floppies, so that they can spread rapidly in computer enthusiast circles.

As the Microsoft Windows increased its dominance in the 1990s with its windows platform and the flexible macros of its applications, it became possible to write malicious code in the macro language of Microsoft Word and similar kind of programs. These macro viruses which were written in the language of word and similar kind of programs infected documents and templates rather than affecting applications (executables), but mostly they relied on the fact that macros in a Word document are also a form of executable code.

Today ,most of the worms written today mainly focus on the Windows OS, although a few like Mare-D and the Lion worm are also written for Linux and UNIX systems. The working of today's worm remains same basic way as 1988's Internet Worm: the network is scanned and the detected vulnerable computers are then leveraged to replicate data . Since they need no human intervention, worms have the ability to spread with incredible speed.

1.2.2. Categorization of Malware

Malware can be categorized as

- **Trojan horses**

  A Trojan horse is a program that attracts the user to run it, concealing a harmful or malicious payload. These are usually present as Keygens, cracks etc for pirated software. The payload executes immediately and can consequentially lead to many undesirable effects, such as deletion of the files or installing additional harmful software. [24]

- **Rootkits**

  Techniques known as remote rootkits allow concealment of the infectious software by modifying the host's operating system so that the said malware is hidden from the user. Rootkits prevent the malicious process from being shown during scans or in the system's list of processes. It may also prevent the user's files from being read. [24]

- **Backdoors**

  A backdoor is a method or software for bypassing normal authentication procedures usually present in private networks. Once the system is compromised, one or more backdoors may be hidden on the system in order to allow easier access in the future. As the name suggests Backdoors may present prior to malicious software on the system, to allow attackers entry. [24]

1.2.3. Anti-Malware Programs

As malware attacks become more frequent, attention has begun to shift from viruses and spyware protection, to malware protection, and programs have been developed specifically to combat them.

Anti-virus and anti-malware software usually bore deep into the operating system's core and/or kernel functions in a similar manner to how malicious software would attempt to operate, though it does this with the user's prior informed permission for protecting the system against attacks. Any time the operating system is inactive, the anti-malware software checks that the OS is performing according to set benchmarks. This decreases the processing speed of the operating system and/or consumes large amounts of system memory. The goal is to pre-empt any illegal operations the malware may attempt to perform on the system, including activities which might leave the system open to bugs or may trigger some unexpected operating system behaviour.

Anti-malware programs can combat malware in two ways:

- Provide real time and D-Day protection against the installation of malicious software and other attacks such as DDOS on a computer. This type of spyware protection works on the same principle that the anti-malware software scans all incoming network data for malware software and blocks any threats it comes across.

- Anti-malware software programs can be used solely for detection and removal of malware software that has already been installed onto a computer. This type of anti-malware software scans the contents of the Windows registry, operating system files, and installed programs on a computer and will provide a list of any threats found, allowing the user to choose which files to delete or keep, or to compare this list to a list of known malware components, removing files that match.

1.3. DeoxyriboNucleic Acid (DNA)

Frederich Miescher was the first scientist to observe DNA in the late 1800s. It wasn't until after more than a century later that researchers unravelled the structure of the DNA molecule and realized it to be the building block of all life.

It was debated for a long time that which was the molecule that actually carried life's biological instructions. Many thought DNA to be too simple a molecule to play such an important role. Instead, they falsely believed that proteins were more likely to carry out this vital function because they were highly complex and available in wider variety of forms.

James Watson's work proved importance of DNA in early 1953 thanks to the work of, Francis Crick, Maurice Wilkins and Rosalind Franklin. By studying X-ray diffraction patterns and then building appropriate models, the scientists were able to figure out the double helix structure of DNA. It was discovered that it is this structure that enables DNA to carry biological information from one generation to the next.

DNA, or deoxyribonucleic acid, is the hereditary material in humans and all other organisms. Every cell in a person's body has the same DNA. DNA is localized in the cell nucleus. Also a small amount of DNA can be found in the mitochondria of the human cell.

1.3.1. Structure of DNA

The term "double-helix" has been used by the scientists to describe the winding, ladder like structure of DNA. This twisted ladder like shape gives DNA the power to pass along genetic instructions with accuracy and precision. Since the chemical pairing have a very specific nature, base A always pairs with base T. similarly C with G. Therefore if one knows the sequence of the bases on one strand of the double-helix, we can the sequence of the bases on the other strand.

DNA as we all know is a polymer type of molecule. Its monomer units are known as a "polynucleotide". Each nucleotide contains a 5-carbon sugar i.e. "deoxyribose", a phosphate group and a nitrogen containing base attached to the sugar. DNA contains four different types of nucleotides which differ only in nitrogenous base. These four nucleotides give one letter abbreviations for the four bases:

- **A** is for Adenine
- **G** is for Guanine
- **C** is for Cytosine
- **T** is for Thymine

Purines i.e. Adenine and guanine are the larger of the two types of bases found in DNA whereas pyrimidines i.e. Cytosine and thymine are smaller. The sequences of these four nucleotides along the backbone encode the actual information.

A special are of the cell called nucleus contains the DNA. Since, the cell is very small, and because organisms have many DNA molecules per cell, packaging mostly done tightly for each DNA molecule and this packaged form is known as chromosome.

During its replication, DNA can be copied because of its unwinding nature. During the various times in the cell cycle, DNA also unwinds so that the instructions contained in it can be used to make proteins for other biological processes. During the process of cell division, DNA in its chromosome for enables the transfer to new cells.

Mitochondria which is responsible for generating the energy which the cell needs to function properly is also the DNA located in the cell structures of the complex organisms like human beings, animals etc.

Two polynucleotide chains, held together by weak thermodynamic forces, form a double stranded macromolecule known as the DNA molecule.

The unique structure of the DNA enables the molecule to copy itself during cell division. During the cell division, the helix of the DNA splits down from the middle into two single strands. These single strands serve as templates for building two new, DNA molecules. During this process, A base is added to T wherever it is present and similarly C adds itself to G and so on until there is a partner for each base.

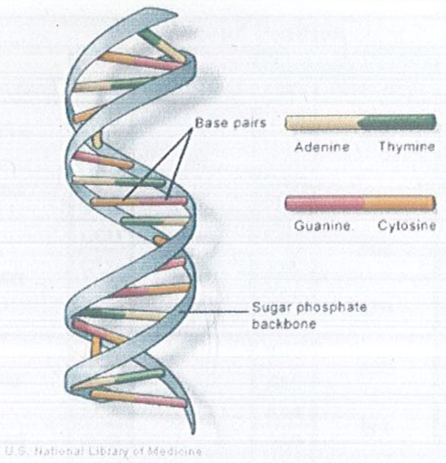


Fig. 1.1 : Structure Of DNA

1.3.2. Proteins and Amino Acids

mRNA which is transcribed from DNA carries information for protein synthesis. There are many techniques which make it a very efficient hence a very good motivation for various cryptography techniques. The various bases of DNA pair with each other to form units called base pairs. These bases are attached to one sugar molecule and one phosphate molecule and together these are known as nucleotides. These nucleotides are the ones responsible for forming the double helical structure of the DNA. In addition, when proteins are made the double helix unwinds to allow a single strand of DNA to serve as a template. This template is then transcribed into mRNA, which is a molecule that conveys vital instructions to the cell's protein making industry.

The information which is stored in the mRNA molecule is then converted into the language understood by the amino acids. This language tells the whole mechanism in which to link the amino acids to produce a new protein to the protein synthesis apparatus.

Three consecutive nucleotides in a DNA molecule are known as codons and these are responsible for encoding an amino acid or stopping a signal for protein synthesis. Because of 4 bases in 3-letter combinations, there are 64 possible codons (4^3 combinations) as depicted in figure 2.

	Second Position								
	U		C		A		G		
First Position	code	Amio Acid	code	Amio Acid	code	Amio Acid	code	Amio Acid	Third Position
U	UUU	phe	UCU	ser	UAU	tyr	UGU	cys	U
	UUC		UCC		UAC		UGC		C
	UUA	leu	UCA		UAA	STOP	UGA	STOP	A
	UUG		UCG		UAG	STOP	UGG	trp	G
C	CUU	leu	CCU	pro	CAU	his	CGU	arg	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	gln	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	ile	ACU	thr	AAU	asn	AGU	ser	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	lys	AGA	arg	A
	AUG	met	ACG		AAG		AGG		G
G	GUU	val	GCU	ala	GAU	asp	GGU	gly	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	glu	GGA		A
	GUG		GCG		GAG		GGG		G

Fig 1.2: Relationship between DNA and encoded peptide

1.4. Discrete Wavelet Transform

Wavelet transform can be defined as the decomposition of a signal in terms of a wavelet. The main idea behind the concept of wavelet transform is basically to take advantage of the correlation structure present inmost real life signals in order to build a sparse approximation. This correlation structure is typically local in space (time) and frequency; neighbouring samples and frequencies are more correlated than those that are far apart. Typical wavelet constructions mainly use the Fourier transformations to build the space-frequency localization.

The popularity of wavelet based transform rose exponentially because of the multi-resolution analysis that it provides. Wavelets can be both orthogonal (orthonormal) or bi-orthogonal. Most commonly used wavelets in watermarking are orthogonal wavelets. On the other hand, Biorthogonality allows the construction of symmetric wavelets and hence linear phase filters.

Let a = {axy} x, y, b = {bxy} x, y i.e., the boldfaced letters will denote the matrix representation of the signals or image which is under consideration. Let U = Vg represent the matrix containing the wavelet coefficients of g, where V is the 2-D dyadic orthogonal wavelet transform operator. It is conventional to denote the subbands of the transform as shown in Fig. 3.

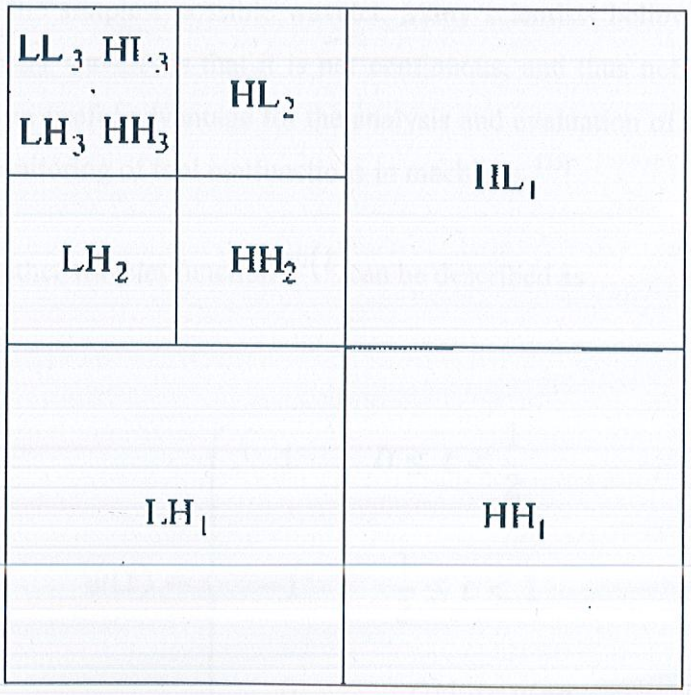


Fig. 1.3: Subbands of 2-D orthogonal wavelet transform

The subbands HH_k, HL_k, LH_k, k = 1,2,…,J are called the details, where k is the scale , with J being the largest (or coarset) scale in the decomposition, and a subband at scale k has size N/2k x N/2k. The subaband LL_J is the low at scale k has size N/2k << N and $N/2^J > 1$. Note that since the transform is the orthogonal, {Vij} are also iid N(0,σ2).

1.5. Haar Wavelet

The HAAR wavelet is a sequence of square-shaped functions which then together form a wavelet family or basis. It is the first and simplest of all in the wavelet family and named after Alfred Haar who proposed it in 1909. He used these functions to give an example of a countable orthonormal system for the space of square-integral functions on the real line.

Properties of Haar like discontinuity and undifferentiability are considered as disadvantageous. These properties can however be an advantageous for analysis of signals with sudden transitions such as monitoring of tool failure in machines. [24]

The Haar wavelet is the simplest possible wavelet. Many scientists believe that the technical disadvantage of the Haar wavelet is that it is not continuous, and thus not differentiable. This property, however, is an useful advantage for the analysis and evaluation of signals with sudden transitions, such as monitoring of tool malfunctions in machines. [25]

The Haar wavelet's mother wavelet function $\psi(t)$ can be described as

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & Otherwise \end{cases}$$

Its scaling function $\phi(t)$ can be described as

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & Otherwise \end{cases}$$

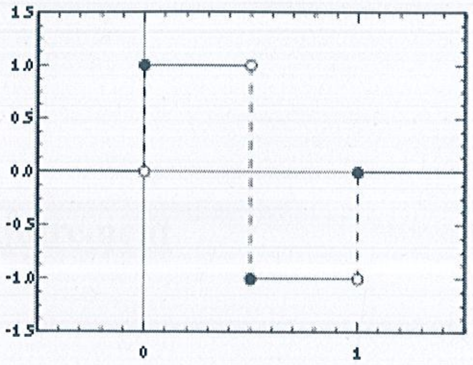


Fig. 1.4: Haar Wavelet

1.6. Aims and Objectives

- We present a new algorithm for securing data that can serve as a tool for maintaining copyright data by adding the owner's signature as required
- It can also be used for securing data and checking authenticity of received data
- In the present situation this application does not have a GUI. But once available it will help users to easily add a watermark. I can be of use to authors selling e-books online, or for researchers publishing papers in journals
- Our aim thus is very clear in building an application for security which provides benefit to the customers as they will not have to run to big clients for securing their data
- The algorithm can be said to "Secure data that may be publically available"

1.7. Resources Requirements

There are two types of requirements in this project:

- *Software Requirements*
- *Hardware Requirements*

1.7.1. Software Requirement

- **Netbeans**
- **Matlab**
- **Java SDK**

1.7.2. Hardware Requirement

Computer system with

- Min 512 Mb RAM
- Bluetooth Connectivity
- Min 1 Gb of free hard disk space

1.8. Scope

In this current internet age when the information is flowing all over the globe in an environment which is not much secure and the flowing information can easily be stolen by some type malicious code. To deal with such problems various techniques have been used like Data Hiding (like Watermarking etc.), Data Encryption and Decryption methods etc. In our project we have enhanced Watermarking (Data Hiding) technique by adding an additional layer of security because of which will become very difficult for the malicious code to attack the data. This technique proposed here can be used:

1. For defence purposes in sharing of some secret data or for hiding and securing some secret information
2. In safeguarding the classified data of any organization for securing passwords etc. from

hackers.

3. For securing any personal data from any type of malicious code.

1.9. Limitations

Limitations of technique can be distributed in various categories:

1. **Scalability:** Since in our program we have kept the size of signature very small, we cannot be sure how it is going to behave when we embed lager signature sizes into the image.

2. **Acceptability:** It is important that both client and manager use the same algorithm otherwise, the client will not be able to decode the message ,

How is our proposed different from other Data Hiding techniques?

We have an added an extra layer of security before watermarking it hence making it more secure. Constraint of resources (we cannot test our program in different environments), so testing will not be much extensive. DNA encoding is not a much used technique hence we do not know about the scalability of our application.

Chapter 2
Project Planning

2.1. UML Diagrams

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Goals of UML

The primary goals in the design of the UML were:

1. To provide end-users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of any single programming languages and only one development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the Object Oriented tools available in the market.
6. Support high-level development and deployment concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

2.1.1. Use Case Diagram

Use case diagrams are important for visualizing, specifying, and documenting the behavior of an element.

- A *use case diagram is a diagram that shows a set of use cases and actors and their* relationships.
- Use case diagrams commonly contain
  - Use cases
  - Actors
  - Dependency, generalization, and association relationships

Description of specific Use Case Diagram:

Actors are *Manager*

Use cases are CONVERT *ARRAY TO IMAGE, ENCODE ARRAY USING DNA TECHNIQUE, CONVERT ENCODED ARRAY TO IMAGE, EMBEDD SIGNATURE TO IMAGE, INPUT IMAGE, INPUT ARRAY, and PROTEIN TABLE*

Dependency or Relationship is <<*uses*>>.

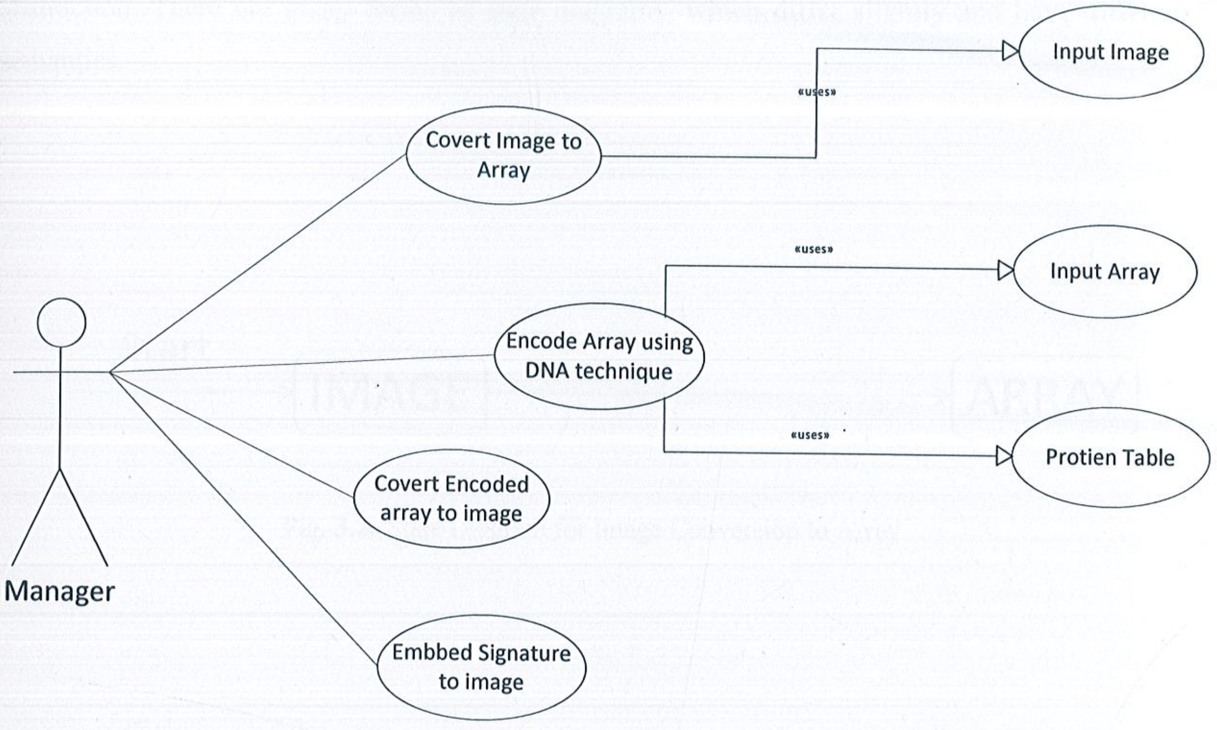

Manager
Covert Image to Array
Input Image
«uses»
Encode Array using DNA technique
Input Array
«uses»
Protien Table
«uses»
Covert Encoded array to image
Embbed Signature to image

Fig. 3.1: Use Case Diagram

2.1.2. State Diagram

A **state diagram** is a type of diagram used in computer science and related fields to describe the behavior of systems. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. There are many forms of state diagrams, which differ slightly and have different semantics.

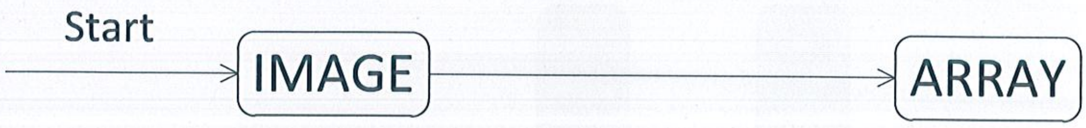


Fig. 3.2: State Diagram for Image Conversion to Array

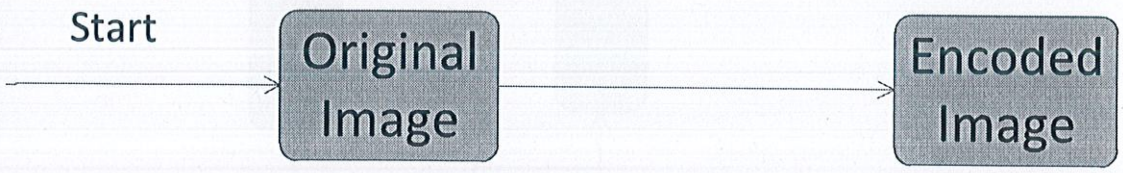


Fig. 3.3: State Diagram of image for embedding signature onto it

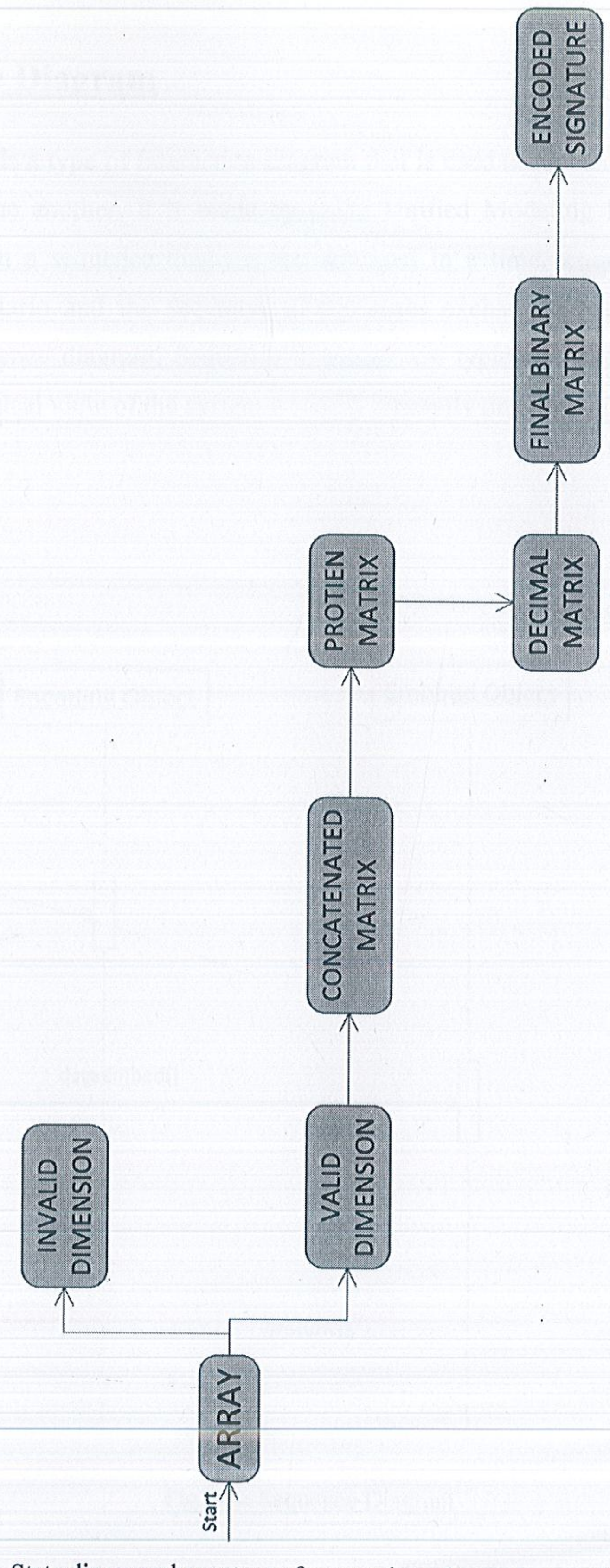


Fig. 3.4: State diagram show steps of conversion of array to encoded array

2.1.3. Sequence Diagram

A sequence diagram is a type of interaction diagram that is used to represent the interoperability of processes with one another. It is made by using Unified Modeling Language(UML). The interactions shown in a sequence diagram are arranged in a time sequence. Objects, classes involved in the scenario and the sequence of messages exchanged between the objects are depicted in the sequence diagram. Sequence diagrams are typically related with the use case realizations in the logical view of the system which is currently under development.

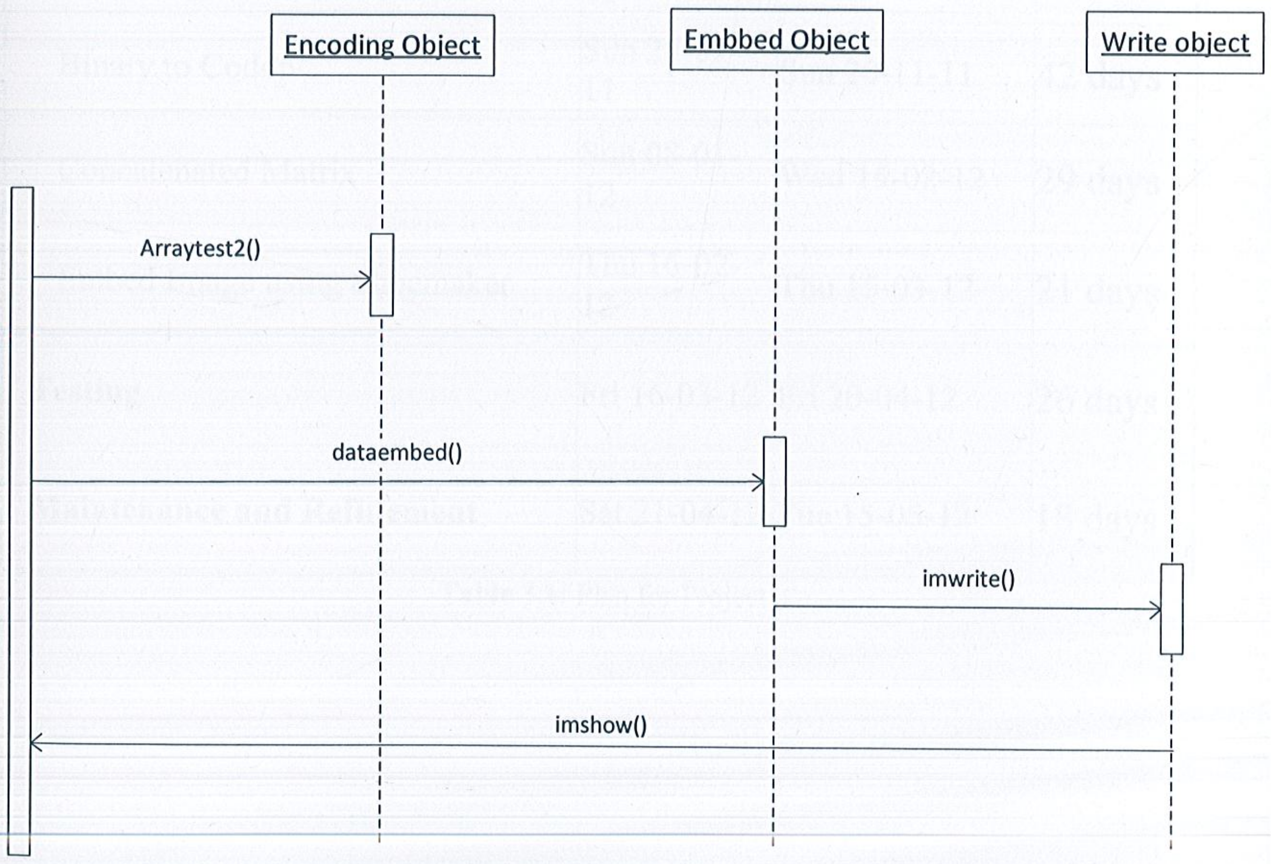


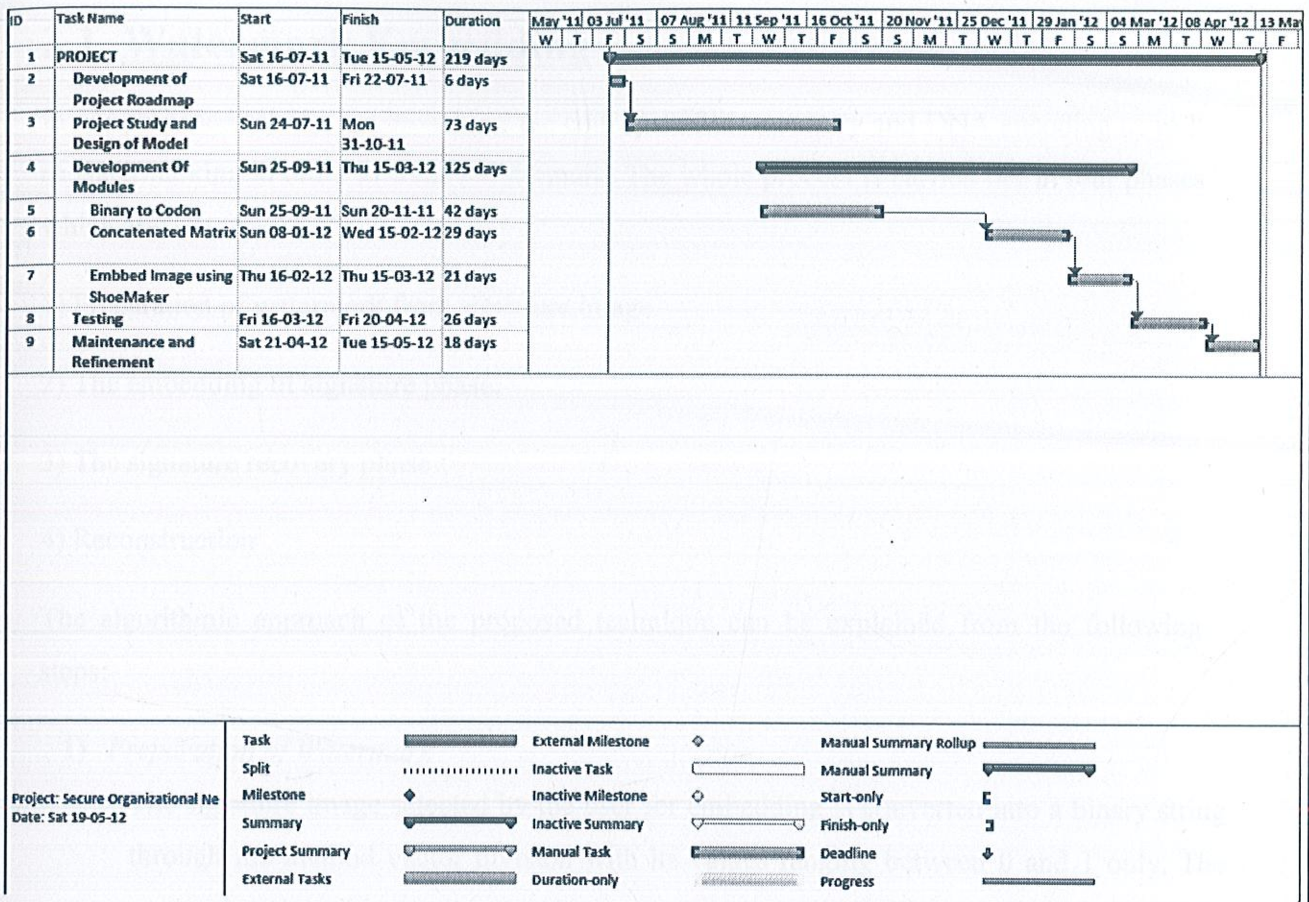
Fig. 3.5: Sequence Diagram

2.2. Plan

Task Name	Start	Finish	Duration
PROJECT	**Sat 16-07-11**	**Tue 15-05-12**	**219 days**
Development of Project Roadmap	Sat 16-07-11	Fri 22-07-11	6 days
Project Study and Design of Model	Sun 24-07-11	Mon 31-10-11	73 days
Development Of Modules	**Sun 25-09-11**	**Thu 15-03-12**	**125 days**
Binary to Codon	Sun 25-09-11	Sun 20-11-11	42 days
Concatenated Matrix	Sun 08-01-12	Wed 15-02-12	29 days
Embed Image using Shoemaker	Thu 16-02-12	Thu 15-03-12	21 days
Testing	Fri 16-03-12	Fri 20-04-12	26 days
Maintenance and Refinement	Sat 21-04-12	Tue 15-05-12	18 days

Table 3.1: Plan for Project

2.3. Gantt Chart



ID	Task Name	Start	Finish	Duration
1	PROJECT	Sat 16-07-11	Tue 15-05-12	219 days
2	Development of Project Roadmap	Sat 16-07-11	Fri 22-07-11	6 days
3	Project Study and Design of Model	Sun 24-07-11	Mon 31-10-11	73 days
4	Development Of Modules	Sun 25-09-11	Thu 15-03-12	125 days
5	Binary to Codon	Sun 25-09-11	Sun 20-11-11	42 days
6	Concatenated Matrix	Sun 08-01-12	Wed 15-02-12	29 days
7	Embbed Image using ShoeMaker	Thu 16-02-12	Thu 15-03-12	21 days
8	Testing	Fri 16-03-12	Fri 20-04-12	26 days
9	Maintenance and Refinement	Sat 21-04-12	Tue 15-05-12	18 days

Project: Secure Organizational Ne
Date: Sat 19-05-12

Fig. 3.6: Gantt Chart

Chapter 3

Algorithm Design

3.1. Watermark Embedding

In our proposed scheme we have used the concepts of bio-memory and DNA encoding as well as watermarking in order to form a watermark. The whole process is carried out in four phases which are:

1) Preparation of watermark from signature image.

2) The embedding of signature phase.

3) The signature recovery phase

4) Reconstruction

The algorithmic approach of the proposed technique can be explained from the following steps:

1) *Preparation of Watermark*

   The signature image selected by the user for embedding is converted into a binary string through the method vector division with its values ranging between 0 and 1 only. The conversion into binary is very important as the whole further processing will be conducted on these two values only.

2) *Encoding the Signature*

   a. The first step is to generate a DNA sequence by converting the binary sequence generated from the above

   b. The sequences obtained from step a.) Is then grouped in three to form tri-nucleotide sequences also known as codons.

   c. Now image regeneration takes place and this is done by again converting codons into binary sequences.

3) *Image Embedding*

a. In order to embed successfully we decompose our input cover image using Discrete Wavelet Transform technique to generate the wavelet coefficients of the input image. Firstly we apply level I DWT on the image so that we get one approximate (LL1) and three detailed (HL1, LH1 and HH1) sub-bands. The coarse region with low frequency coefficients is represented by low frequency coefficients. For the process of embedding the detailed coefficients (HL1, LH1) are used.

b. To ensure confidentiality and for watermarking we use a key which generates pseudorandom sequences, these sequences are equal to the number of co-efficient used for embedding.

c. We generate matrices of PN sequences equal to the number of co-efficients, having variance and mean 1 and 0 respectively using a secret key. These matrices should be of the same size as that of the initial matrix. This secret key is shared between the embedder and the authenticator.

d. The pseudo-random sequences thus generated are inserted into the horizontal wavelet coefficients when the watermark bit is zero, this is done to differentiate between watermark '0' and '1'.In order to optimize the robustness of the watermark , a large embedding factor may be used and this is done at the expense of host fidelity. For our project, we use the embedding factor between 0.5 and 0.7 in order to obtain a balanced mix of robustness and fidelity.

e. We then update the horizontal detailed coefficients with the help of pseudo-random sequences and using Cox's algorithm only for '0' Watermark bit.

f. In the final step, we reconstruct our image using inverse DWT.

4) *Pseudocode for watermark embedding:*

```
    begin
convert offline handwritten signature image to binary array a
    for i:=every element of a
            c[s]:= a[i]*10 + a[i+1];
                    i:=i+1;
```

```
                s:=s+1;
        end for
        for j:=every element in c
                        p[t]:= corresponding code( a, g, c or t);
                n[t]:= respective numeral code( 0, 1, 2 or  3);
        j:=j+1;
        t:=t+1;
        end for
        n[t]:=0
    if ((length of a\2) mod 3 == 1)
        n[t+1]:=0
            for k:=every three elements in n
            pro[u]:=convert from base 4 to base 10;
                u:=u+1;
        len:= length of pro;
      end for
        for l:=0 to len
        f:=convert pro[v] to binary;
        z:=6-(length of f);
        if(z!=0)
      enter z zeroes at the beginning of binary string    end if
    final[l]=f;
    end for
embed final code in image
end
```

3.2. Watermark Extraction

The following steps depict the watermark extraction phase:

a) The first step is to regenerate the pseudo-random sequences and this is done with the help of the sharing of secret key.

b) We generate a sequence of only 1's equal to the length of the original watermark. This sequence is then used to reconstruct the watermark.

c) In order to obtain the horizontal detailed coefficients We perform DWT of the watermarked image.

d) We calculate the correlation between the horizontal and PN sequence thus generated and stored in a 1-D sequence equal to the length of the watermark.

e) We then calculate the Standard deviation by using the correlation sequence thus generated and then compare it to each value of the sequence in order to decide the watermark bit.

f) The last step is to update the watermark bit depending on the values obtained from the above steps.

STEP 1: Pseudocode for watermark extraction:

```
start
apply DWT to the image;
obtain CH' and CV';
 for c= each block in HL
h=correlation with CH';
 v= correlation with CV';
```

```
      if (h>v)
          a[c]=0;
      else
          a[c]=1;
      end for
        for i:=every six elements in the array a
        for k:=every element of p
                c[l]:=p[k]\16;
                d:=p[k] mod 16;
          c[l+2]:=d mod 4;
          l:=l+3;
          end for
          Len: = length of array c;
  p[j]:= convert binary to decimal;
                          j:=j+1;
          end for
  endif
  c[l+1]:=d\4;
  while (n<lenf)
  f[n]:=assign respective codes to c[n]; (a, g, c and t)
                final[u]:=c[n]\2;
                final[u+1]:=c[n] mod 2;
```

```
            n:=n+1;
        u:=u+2;
        end while
print final;
```

Step 2: DNA Decoding (Template Matching)

a. The first step is to make the groups of six elements of the watermark so that the further encoding process can be performed.
b. We then convert each group in decimal representation.
c. We then separate codons (i.e. tri-nucleotide sequences) to form binary sequences.
d. Convert the binary sequence into the signature format.
e. The last step is to output the signature.

Chapter 4

Testing

4.1. The Goal of Testing

In different publications, the definition of testing varies according to the purpose, process, and level of testing described. Miller gives a good description of testing.

The general aim of testing is to affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances.

The most important will be performance testing as when so many users will be simultaneously using the application and the database will be accessed as such frequent high rates it is must that performance should not deteriorate.

4.2. Significance Measures

To compare the changes between the original image and the image with the embedded watermark we use two measures.

4.2.1. Peak Signal to Noise Ratio (PSNR)

The phrase peak signal-to-noise ratio, hereby referred to as PSNR, is an term used in the engineering field as it is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the accuracy of its representation. Because many signals may have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale.

We mostly use PSNR as a measure of quality of reconstruction of lossy compression codecs (e.g., for image compression). The signal in this case is the original data, and the noise is the error introduced by compression. Comparison of compression codecs helps us to get an approximation of human perception of reconstruction quality, therefore in some test cases one representation may appear to be closely related to the original than other; even though value of PSNR is lower (a higher PSNR would normally indicate that the reconstruction is of higher quality).

For testing purposes the Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) were calculated for benchmarking which have been given below.

$$MSE = \frac{1}{mn}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1}(I(x,y) - W(x,y))^2$$

where I and W are the original and the watermarked images having a resolution of m*n.

$$PSNR = 10\log_{10}\frac{max^2}{MSE}$$

4.2.2. Structural Similarity Index Measure (SSIM)

The Structure similarity index (SSIM) is basically a method for calculating the similarity between any two images. This method calculates the image quality taking the initial uncompressed or distortion-free image as reference. SSIM is designed to overcome the shortcomings of some traditional methods like that of peak signal-to noise ratio(PSNR) and mean squared error(MSE), which have proven to be not much consistent with human eye perception.

SSIM is based on the proven hypothesis that HVS is highly adapted for extracting the structural information. The SSIM works by measuring structural similarity local patterns of pixel intensities that have been normalized for luminance and contrast. The formula for calculating SSIM (MSSIM) index is as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_1)}{({\mu_x}^2 + {\mu_y}^2 + C_1)({\sigma_x}^2 + {\sigma_y}^2 + C_1)}$$

$$MSSIM(x,y) = \frac{1}{M}\sum_{m=1}^{M} SSIM(x_m, y_m)$$

4.3. The Testing Spectrum

Testing is involved in every stage of our Algorithm, but the testing done at each level of software development is different in nature and has different objectives.

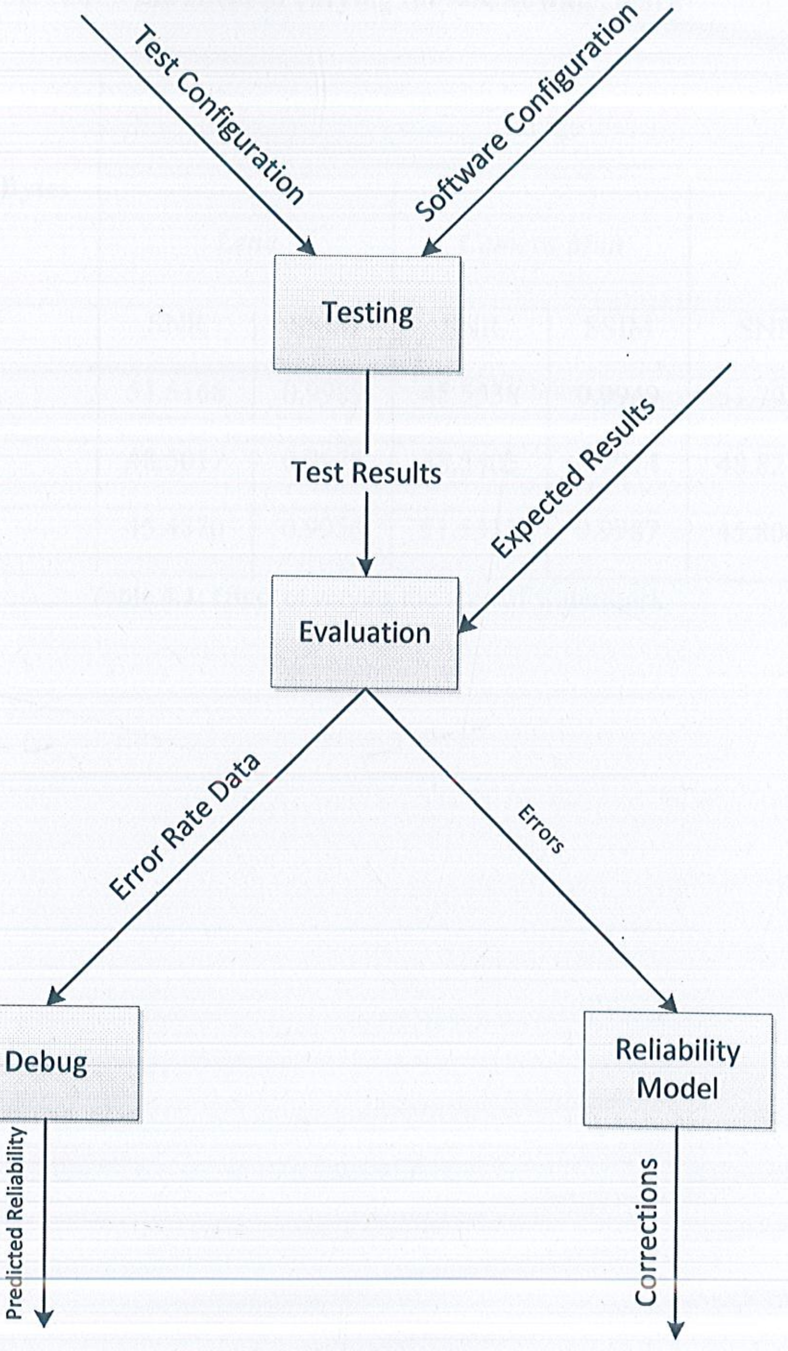


Fig. 4.1: Testing Information Flow

The final testing involves encoding of different sized signatures onto selected images. These images were Lenna, Boat and Cameraman. The measures used for testing of these images were SSIM and SNR as discussed above.

The Following table shows the effect of varying the size of watermark

Message size in Bytes	**Image**					
	Lena		***Camera Man***		***Boat***	
	SNR	SSIM	SNR	SSIM	SNR	SSIM
1094	51.5168	0.9989	45.5538	0.9949	51.7956	0.9993
1142	48.5017	0.9978	48.5403	0.9974	48.8217	0.9986
1334	45.4870	0.9956	51.5332	0.9987	45.8047	0.9971

Table 4.1: Effect of varying the size of watermark

4.4. For 4px Encoded Signature

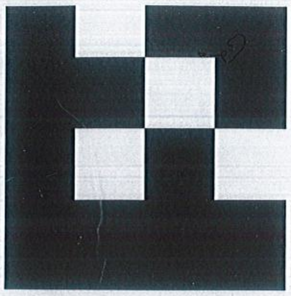


Fig. 4.2: Encoded Signature (4px)

Watermarked Image (Lena)

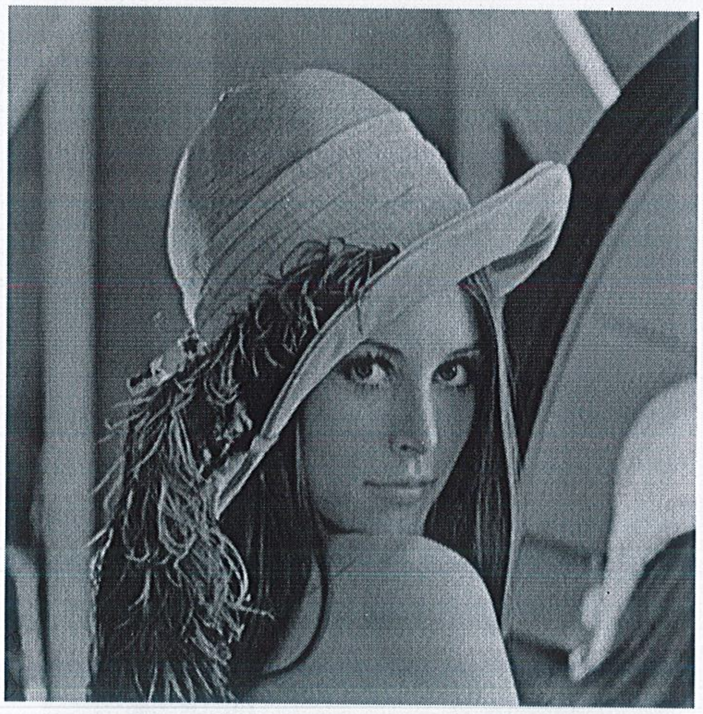


Fig. 4.3: Lena (Watermarked, 4px Signature)

Watermarked Image (Boat)

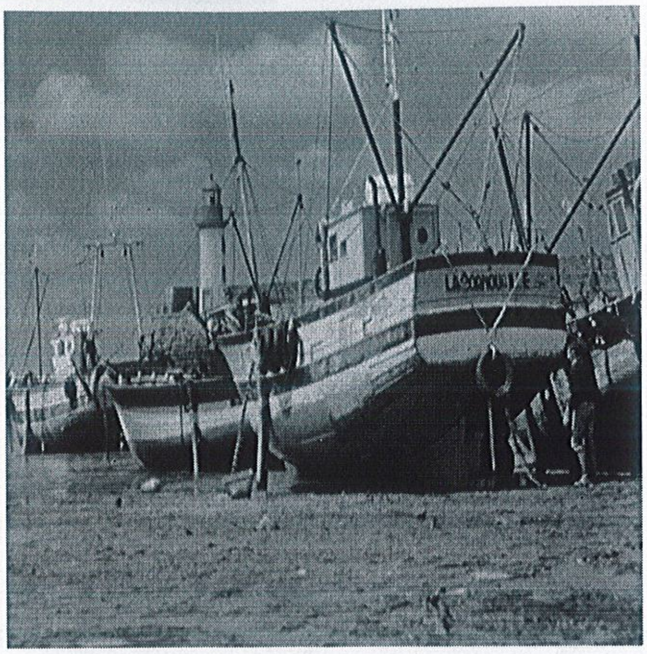


Fig. 4.4: Boat (Watermarked, 4px Signature)

Watermarked Image (Cameraman)

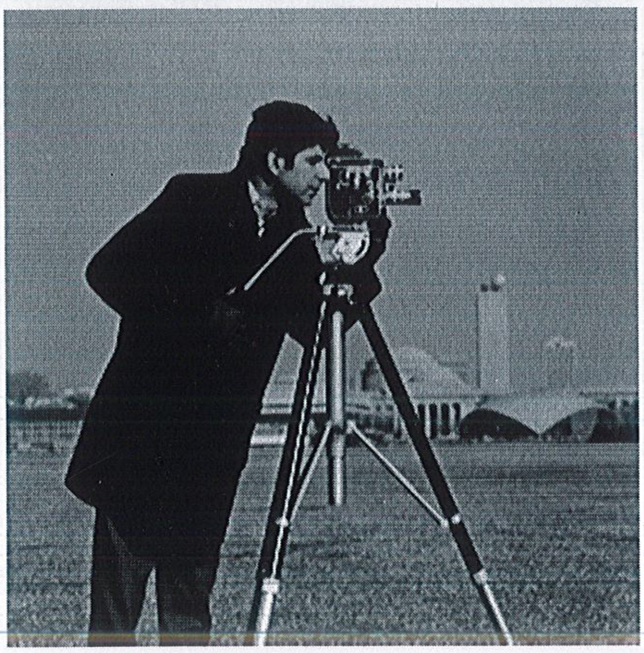


Fig. 4.5: Cameraman (Watermarked, 4px Signature)

4.5. <u>For 8px Encoded Signature</u>

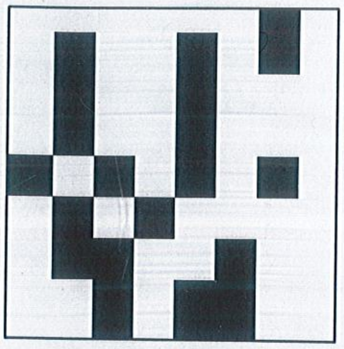


Fig. 4.6: Encoded Signature (8px)

<u>Watermarked Image (Lena)</u>

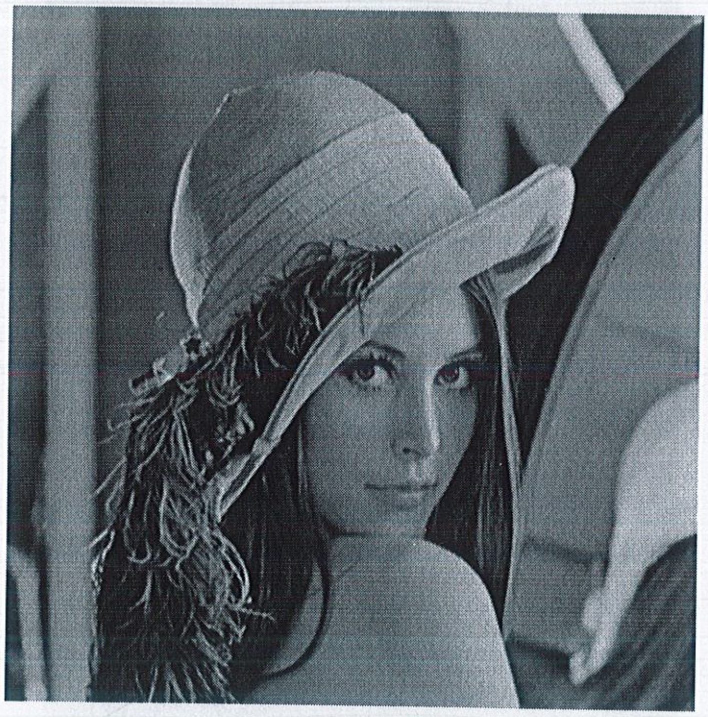


Fig. 4.7: Lena (Watermarked, 8px Signature)

Watermarked Image (Boat)

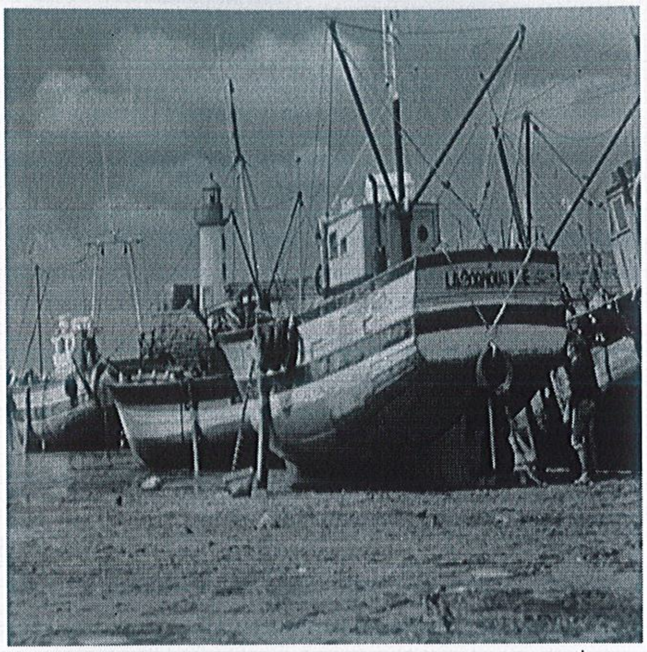


Fig. 4.8: Boat (Watermarked, 8px Signature)

Watermarked Image (Cameraman)

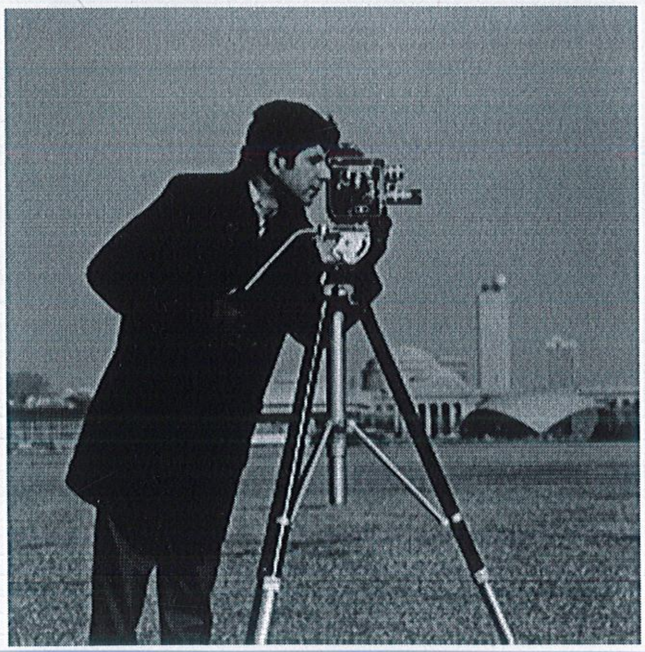


Fig. 4.9: Cameraman (Watermarked, 8px Signature)

4.6. For 16px Encoded Signature

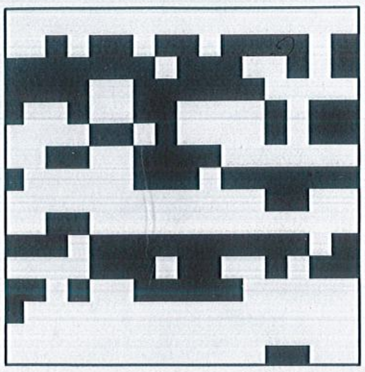


Fig. 4.10: Encoded Signature (16px)

Watermarked Image (Lena)

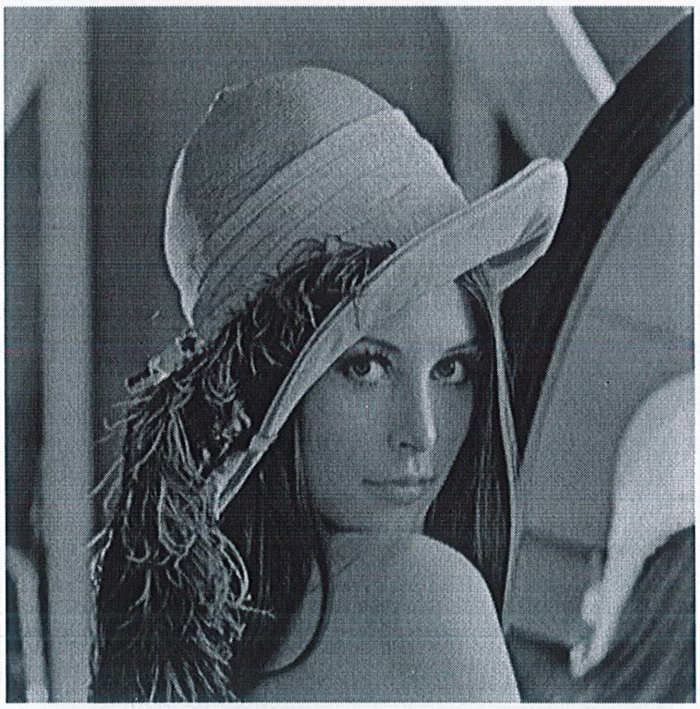


Fig. 4.11: Lena (Watermarked, 16px Signature)

Watermarked Image (Boat)

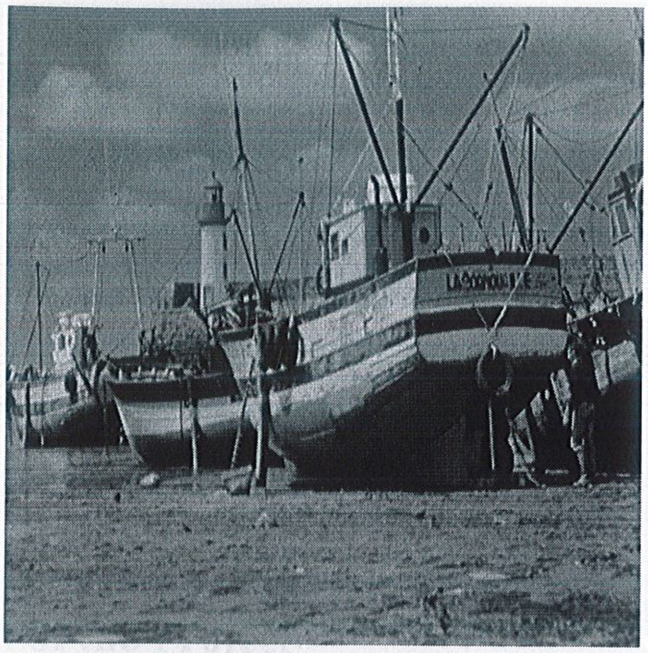


Fig. 4.12: Boat (Watermarked, 16px Signature)

Watermarked Image (Cameraman)

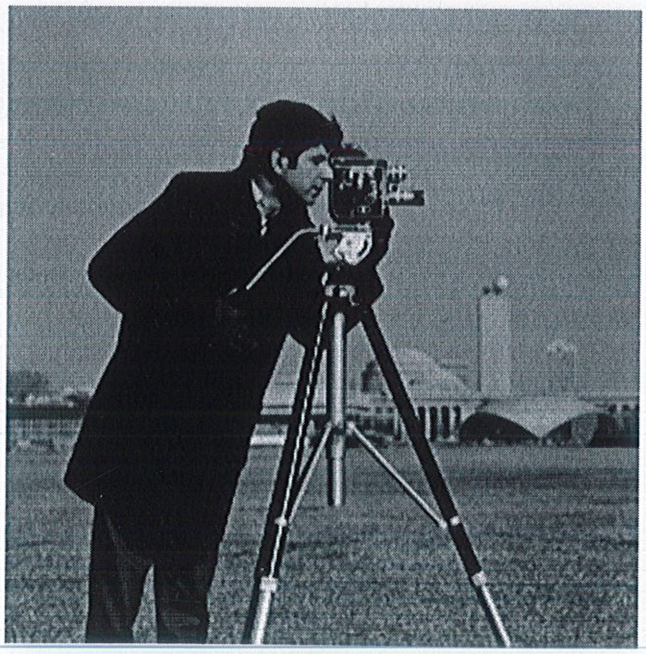


Fig. 4.13: Cameraman (Watermarked, 16px Signature)

Chapter 5

Conclusion

5.1 Results and Conclusion

We present a new approach towards encryption and security of organizational data from malicious codes. This algorithm can serve as a basic building block for development of new applications and more secure methods of encoding, encryption and securing information. A new encoding technique is used comprising the knowledge of Bio memory and DNA encoding using Codons.

5.2 Contribution of the Project

We have thus presented a novel method of encoding data into images. This method combines the fields of biology and Computer science to give way to a new era of encoding techniques. This method adds an additional layer of security in the form of DNA encoding matrix. Such Additional layer of security is not usually present in current standards.

Effectively it will change the look of the industry if it's successful which can be made possible by the way industry look towards it. This technique is a thing of the future and once in existence, it will change the look of the industry.

References

[1]Clelland C, Risca V, Bancroft C. "Hiding messages in DNA microdots". Nature. 1999;399:533–534.

[2]Arita M, Ohashi Y." Secret signatures inside genomic DNA". Biotechnol Prog. 2004;20:1605–1607.

[3]Hayam Mousa, Kamel Moustafa, Waiel Abdel-Wahed, and Mohiy Hadhoud. "Data Hiding Based on Contrast Mapping Using DNA Medium" , The International Arab Journal of Information Technology, Vol. 8, No. 2, April 2011.

[4] ElGamal T., "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," Computer Journal of IEEE Transactions on Information Theory, vol. 31, no. 4, pp. 469-472, 1985.

[5] Rijmen P., "Advanced Encryption Standard," in Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology, pp. 19-22, 2001.

[6] Rivest L., Shamir A., and Adleman L., "A Method for Obtaining Digital Signature and Public Key Cryptosystem," Computer Journal of Communications of the ACM, vol. 21, no. 2, pp. 120-126, 1978.

[7]Smid E. and Branstad M., "Data Encryption Standard," in Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology, pp. 550-559, 1988.

[8] Hwang J. and Chang C., "Hiding a Picture in Two Pictures," Computer Journal of Optical Engineering, vol. 40, no. 3, pp. 342-351, 2001.

[9] Lin H., Hu C., and Chang C., "Both Color and Gray Scale Secret Images Hiding in a Color Image," Computer Journal of International Journal on Pattern Recognition and Artificial Intelligence, vol. 16, no. 6, pp. 697-713, 2002.

[10]Saeb M., El-abd E., and El-Zanaty M., "On Covert Data Communication Channels Employing DNA Recombinant and Mutagenesisbased Steganographic Techniques," Computer Journal of BioSystems, vol. 57, no. 2, pp. 13-22, 2000.

[11] Leier A., Richter C., Banzhaf W., and Rauhe H., "Cryptography with DNA Binary Strands," Computer Journal of BioSystems, vol. 57, no. 2, pp. 13-22, 2000.

[12] Gehani A., Labean T., and Reif J., "DNA Based Cryptography," Computer Journal of IMACS DNA-Based Computer, American Mathmatical Society, USA, vol. 2950, no. 456, pp. 34-50, 2004.

[13] Hartung F, Kutter M, "Multimedia watermarking Technique", IEEE proceeding on Signal Processing", vol. 87, no..7, pp.1079-1107,July 1999.

[14]Anil K. Jain, Stan Z.Li,"Encyclopedia of Biometrics"Springer(2009).

[15]F. Bartolini , A. Tefas , M. Barni and I. Pitas , "Image Authentication Techniques for Surveillance Applications", IEEE proceedings , Vol .89, No. 10 , October 2001.

[16]Wang, Z., Bovik, Alan C., Sheikh, Hamid R., Simoncelli, Eero P.: Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions On Image Processing, vol. 13, no. 4, (2004).

[17]S. Grace Chang, Student Member, IEEE, Bin Yu, Senior Member, IEEE, and Martin Vetterli, Fellow, IEEE, Adaptive Wavelet Thresholding for Image Denoising and Compression, IEEE Transactions On Image Processing, VOL. 9, NO. 9, September 2000.

[18] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," IEEE Trans. Pattern Anal. Machine Intell., vol. 11, pp. 674–693, July 1989.

[19] M. Vetterli and J. Kova˘cevic´, Wavelets and Subband Coding. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[20] Symantec Internet Security Threat Report: Trends for July–December 2007 (Executive Summary). XIII. Symantec Corp.. April 2008. p. 29.

[21] "F-Secure Reports Amount of Malware Grew by 100% during 2007" (Press release). F-Secure Corporation. December 4, 2007.

[22] Meenakshi S Arya, Nikhita Jain, Jai Sisodia, Nukul Sehgal, "DNA Encoding Based Feature Extraction for Biometric Watermarking",2011 International Conference on Image Information Processing (ICIIP 2011)

[23] An Undirected Attack Against Critical Infrastructure, vol. 1-2 http://www.uscert.gov/control_systems/pdf/undirected_attack0905.pdf, pg. 11

[24] http://en.wikepedia.org, Wikipedia Online Encyclopedia

[25] Lee, B.; Tarng, Y. S. (1999). "Application of the discrete wavelet transform to the monitoring of a tool in end milling using the spindle motor current". *International Journal of Advanced Manufacturing Technology* 15 (4): 238–243.

Appendix

CODE

DNA Encoding

```
import java.util.Scanner;

public class ArrayTest2

{

public static void main(String args[])

{

Scanner s= new Scanner(System.in);

int i,j,x,y;

System.out.println("enter the value of n");

//Enter values of  n to form nxn matrix

int n=s.nextInt();

int a[][]=new int[n][n];

int c[][]=new int[n][n];

System.out.println("enter the array elements of matrix:");

// Enter elements of matrix

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

{

a[i][j]=s.nextInt();

}

}

//end of a input matrix

try

{

for(i=0;i<n;i++)

{
```

```
for(j=0;j<n;j++)
{
x =       a[i][j];
y = a[i][j+1];
c[i][j] = (10*x) + (1*y);
j=j+1;
}
}
// end of concatenated matrix
System.out.println("Concatenated Matrix :- \n");
for(i=0;i<c.length;i++)
{
for(j=0;j<(c[i].length);j++)
{
        System.out.println(c[i][j]);
j=j+1;
}
}
//concatenation matrix c
int k =0;
String d[]=new String[c.length*c.length/2];
int de[]=new int[c.length*c.length/2];
for(i=0;i<c.length;i++)
{
for(j=0;j<c[i].length;j++)
{
int z = c[i][j];
System.out.print("z slection \t");
System.out.print(z+"\n");
switch (z){
case 0:d[k]="A";de[k]=0;
```

```
   System.out.println("Out:- A");
   break ;
  case 1:d[k]="T";de[k]=1;
   System.out.println("Out:- T");
   break;
  case 10:d[k]="G";de[k]=2;
   System.out.println("Out:- G");
   break;
  case 11:d[k]="C";de[k]=3;
   System.out.println("Out:- C");
   break;
  }
  j=j+1;
  k++;
  }
  }
           System.out.println("protein matrix");
  for (k=0;k<d.length;k++)
  {
           System.out.println(d[k]);
  }
  // protein matrix completed
  //allotting combinations of 3-3 ATGC to decimal no.
  //assuming A-0, T-1, G-2, C-3
  System.out.println("Corresponding Decimal matrix(assuming A-0, T-1, G-2, C-3)");
  for (k=0;k<d.length;k++)
  {
           System.out.println(de[k]);
  }
  // corresponding decimal matrix completed
  int m=0;
```

```
int out[] = new int[(de.length/3)+(de.length%3)];
if (de.length%3 == 1)
{
        for(k=0;k<de.length-1;k=k+3)
        {
                out[m] = de[k] + (4*de[k+1])+ (16*de[k+2]);
                m++;
        }
        for(k=de.length-1;k<de.length+2;k=k+3)
        {
                out[m] = de[k] + (4*0)+ (16*0);
                m++;
        }
}
else if(de.length%3 == 2)
{
        for(k=0;k<de.length-2;k=k+3)
        {
                out[m] = de[k] + (4*de[k+1])+ (16*de[k+2]);
                m++;
        }
        for(k=de.length-2;k<de.length+1;k=k+3)
        {
                out[m] = de[k] + (4*de[k+1])+ (16*0);
                m++;
        }
}
System.out.println("corresponding protein decimal matrix");
for(k=0;k<out.length;k++)
{
        System.out.println(out[k]);
```

```
}
// corresponding protein decimal matrix completed
String pro[] = new String[out.length];
String bin[]= new String[out.length];
k=0;
for(m=0;m<out.length;m++)
{
String z = "";
        switch (out[m]){
        case 0:pro[k]="0";
         break ;
        case 1:pro[k]="1";
         break;
        case 2:pro[k]="2";
         break;
        case 3:pro[k]="3";
         break;
        case 4:pro[k]="4";
         break ;
        case 5:pro[k]="5";
         break;
        case 6:pro[k]="6";
         break;
        case 7:pro[k]="7";
         break;
        case 8:pro[k]="8";
         break ;
        case 9:pro[k]="9";
         break;
        case 10:pro[k]="10";
         break;
```

Annexure C

- Internal Roads with retaning wall and metalling
- Parks & Lawns
- Plot Landscaping, Development & Retaining Walls
- Water Storage Tanks
- Tube Well
- Water Supply Line
- Main Gate with reception Lounge
- Wired Fencing
- Sewerage System
- Underground Electricity & Power Cables Supply Line
- Common Parking
- Rain Water Harvesting Tank
- Plantation
- Swimming Pool
- Fountain
- Amphitheater
- Club House

```
case 11:pro[k]="11";
 break;
case 12:pro[k]="12";
 break;
case 13:pro[k]="13";
 break;
case 14:pro[k]="14";
 break;
case 15:pro[k]="15";
 break;
case 16:pro[k]="16";
 break;
case 17:pro[k]="17";
 break;
case 18:pro[k]="18";
 break;
case 19:pro[k]="19";
 break;
case 20:pro[k]="20";
 break ;
case 21:pro[k]="21";
 break;
case 22:pro[k]="22";
 break;
case 23:pro[k]="23";
 break;
case 24:pro[k]="24";
 break ;
case 25:pro[k]="25";
 break;
case 26:pro[k]="26";
```

```
break;
case 27:pro[k]="27";
break;
case 28:pro[k]="28";
break ;
case 29:pro[k]="29";
break;
case 30:pro[k]="30";
break;
case 31:pro[k]="31";
break;
case 32:pro[k]="32";
break ;
case 33:pro[k]="33";
break;
case 34:pro[k]="34";
break;
case 35:pro[k]="35";
break;
case 36:pro[k]="36";
break ;
case 37:pro[k]="37";
break;
case 38:pro[k]="38";
break;
case 39:pro[k]="39";
break;
case 40:pro[k]="40";
break ;
case 41:pro[k]="41";
break;
```

```
case 42:pro[k]="42";
 break;
case 43:pro[k]="43";
 break;
case 44:pro[k]="44";
 break ;
case 45:pro[k]="45";
 break;
case 46:pro[k]="46";
 break;
case 47:pro[k]="47";
 break;
case 48:pro[k]="48";
 break ;
case 49:pro[k]="49";
 break;
case 50:pro[k]="50";
 break;
case 51:pro[k]="51";
 break;
case 52:pro[k]="52";
 break ;
case 53:pro[k]="53";
 break;
case 54:pro[k]="54";
 break;
case 55:pro[k]="55";
 break;
case 56:pro[k]="56";
 break;
case 57:pro[k]="57";
```

```
            break;
          case 58:pro[k]="58";
            break ;
        case 59:pro[k]="53";
            break;
          case 60:pro[k]="60";
            break;
          case 61:pro[k]="61";
            break;
          case 62:pro[k]="62";
            break;
          case 63:pro[k]="63";
            break;
          case 64:pro[k]="64";
            break ;
          }
z = Integer.toBinaryString(out[m]);
if(z.length()<6);
{
          for(int p = z.length();p<6;p++)
                    {
                              z = "0" + z;
                    }
}
bin[k] = z;
k++;
}//corresponding protein names allocation and binary matrix compilation completed
System.out.println("Final Binary matrix");
for(k=0;k<out.length-1;k++)
{
          System.out.println(bin[k]);
```

```
}//final binary matrix
}
catch(Exception e)
{
}
}//end of main
}//end of class
```

Watermark Embedding

```
clear all;

clc;

close all;


% save start time

start_time=cputime;


k= 0.5;            % set the gain factor for embeding


% read in the cover object

file_name='lena.bmp';

cover_object=double(imread(file_name));


% determine size of watermarked image

Mc=size(cover_object,1);  %Height

Nc=size(cover_object,2);   %Width


% read in the message image and reshape it into a vector

file_name='encoded4px.bmp';

message=double(imread(file_name));


 Mm=size(message,1);                          %Height

Nm=size(message,2);                           %Width

%message_vector=round(reshape(message,Mm*Nm,1)./256);

%message_vector= [ 1 1 1 1 1 1 0 1

%1 0 1 1 0 1 0 1

%1 0 1 1 0 1 1 1

%1 0 1 1 0 1 1 1

%0 1 0 1 0 1 0 1

%1 0 1 0 1 1 1 1
%1 0 0 1 1 0 1 1
%1 1 0 1 0 0 1 1]

message_vector = message;

% read in key for PN generator
file_name='_key.bmp';
key=double(imread(file_name))./256;

% reset MATLAB's PN generator to state "key"
%rand('state',key);

[cA1,cH1,cV1,cD1] = dwt2(cover_object,'haar');

% add pn sequences to H1 and V1 componants when message = 0
for (kk=1:length(message_vector))
  pn_sequence_h=round(2*(rand(Mc/2,Nc/2)-0.5));
  pn_sequence_v=round(2*(rand(Mc/2,Nc/2)-0.5));

  if (message(kk) == 0)
     cH1=cH1+k*pn_sequence_h;
     cV1=cV1+k*pn_sequence_v;
  end
end

% perform IDWT
watermarked_image = idwt2(cA1,cH1,cV1,cD1,'haar',[Mc,Nc]);

% convert back to uint8
watermarked_image_uint8=uint8(watermarked_image);
```

```
% write watermarked Image to file
imwrite(watermarked_image_uint8,'DONE.bmp','bmp');

% display processing time
elapsed_time=cputime-start_time,

% calculate the PSNR
snr=snr(cover_object,watermarked_image)
ssim = ssim_index(cover_object,watermarked_image)

% display watermarked image
figure(1)
imshow(watermarked_image_uint8,[])
title('Watermarked Image')
```

Calculation od SNR

```
% Calcution of SNR
function a = snr(im,x)


add=0;add1=0;
%im=im2double(im);
%x=im2double(x);
sub=imsubtract(im,x);
[m n]=size(im);
for i=1:m
  for j=1:n
     c=power(x(i,j),2);
     add=add+c;
     d=power(sub(i,j),2);
     add1=add1+d;
  end
end
a=10*log10(add/add1);
```

SSIM Index

```
function [mssim, ssim_map] = ssim_index(img1, img2, K, window, L)


if (nargin < 2 | nargin > 5)
   mssim = -Inf;
   ssim_map = -Inf;
   return;
end


if (size(img1) ~= size(img2))
   mssim = -Inf;
   ssim_map = -Inf;
   return;
end


[M N] = size(img1);


if (nargin == 2)
  if ((M < 11) | (N < 11))
            mssim = -Inf;
            ssim_map = -Inf;
   return
  end
  window = fspecial('gaussian', 11, 1.5);      %
  K(1) = 0.01;                                                  % default settings
  K(2) = 0.03;                                                  %
  L = 255;                 %
end


if (nargin == 3)
  if ((M < 11) | (N < 11))
```

```
                mssim = -Inf;
                ssim_map = -Inf;
          return
        end
        window = fspecial('gaussian', 11, 1.5);
        L = 255;
        if (length(K) == 2)
          if (K(1) < 0 | K(2) < 0)
                         mssim = -Inf;
                       ssim_map = -Inf;
                       return;
          end
        else
                  mssim = -Inf;
               ssim_map = -Inf;
                  return;
        end
      end

      if (nargin == 4)
        [H W] = size(window);
        if ((H*W) < 4 | (H > M) | (W > N))
                 mssim = -Inf;
                 ssim_map = -Inf;
          return
        end
        L = 255;
        if (length(K) == 2)
          if (K(1) < 0 | K(2) < 0)
                       mssim = -Inf;
                     ssim_map = -Inf;
```

```
                        return;
        end
      else
                mssim = -Inf;
              ssim_map = -Inf;
              return;
      end
   end

   if (nargin == 5)
     [H W] = size(window);
     if ((H*W) < 4 | (H > M) | (W > N))
              mssim = -Inf;
              ssim_map = -Inf;
       return
     end
     if (length(K) == 2)
       if (K(1) < 0 | K(2) < 0)
                       mssim = -Inf;
                     ssim_map = -Inf;
                     return;
      end
    else
              mssim = -Inf;
            ssim_map = -Inf;
              return;
    end
  end

  C1 = (K(1)*L)^2;
  C2 = (K(2)*L)^2;
```

```
 window = window/sum(sum(window));
 img1 = double(img1);
 img2 = double(img2);

 mu1   = filter2(window, img1, 'valid');
 mu2   = filter2(window, img2, 'valid');
 mu1_sq = mu1.*mu1;
 mu2_sq = mu2.*mu2;
 mu1_mu2 = mu1.*mu2;
 sigma1_sq = filter2(window, img1.*img1, 'valid') - mu1_sq;
 sigma2_sq = filter2(window, img2.*img2, 'valid') - mu2_sq;
 sigma12 = filter2(window, img1.*img2, 'valid') - mu1_mu2;

 if (C1 > 0 & C2 > 0)
   ssim_map = ((2*mu1_mu2 + C1).*(2*sigma12 + C2))./((mu1_sq + mu2_sq + C1).*(sigma1_sq + sigma2_sq + 
 C2));
 else
   numerator1 = 2*mu1_mu2 + C1;
   numerator2 = 2*sigma12 + C2;
         denominator1 = mu1_sq + mu2_sq + C1;
  denominator2 = sigma1_sq + sigma2_sq + C2;
  ssim_map = ones(size(mu1));
  index = (denominator1.*denominator2 > 0);
  ssim_map(index) = (numerator1(index).*numerator2(index))./(denominator1(index).*denominator2(index));
  index = (denominator1 ~= 0) & (denominator2 == 0);
  ssim_map(index) = numerator1(index)./denominator1(index);
end

mssim = mean2(ssim_map);

return
```