IMAGE ENCRYPTION USING MERKLE HELLMAN KNAPSACK CRYPTOSYSTEM

A project report submitted for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

MUMMANA VINOD KUMAR	**1210311234**
KOLLIPARA VAIBHAV	**1210311228**
MARRIPATI SRIKRISHNA KARTEEK	**1210311230**
YELLAPU BHARGAVA RAJESH	**1210311266**

Under the esteemed guidance of

Ms. J. Hyma
Assistant Professor, Dept. of CSE
GITAM University, Visakhapatnam

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY
GITAM Institute of Technology, GITAM University
Visakhapatnam-530045

GITAM INSTITUTE OF TECHNOLOGY
GITAM UNIVERSITY, RUSHIKONDA,
VISAKHAPATNAM-530045
(2011-2015)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled, "IMAGE ENCRYPTION USING MERKLE HELLMAN KNAPSACK CRYPTOSYSTEM" is being submitted by MUMMANA VINOD KUMAR, KOLLIPARA VAIBHAV, MARRIPATI SRIKRISHNA KARTEEK, YELLAPU BHARGAVA RAJESH in partial fulfillment of the academic requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering to GITAM University, is a record of bonafide work carried out by them under my guidance and supervision. The results obtained in the project have not been submitted to any other industry or institute for the award of any degree.

Project Guide

Ms. J. Hyma

Assistant Professor
GITAM University
Visakhapatnam

Head of the Department

Dr. P. V. Nageswara Rao

Professor, Dept. of CSE
GITAM University
Visakhapatnam

DECLARATION

We hereby declare that the project entitled "**IMAGE ENCRYPTION USING MERKLE HELLMAN KNAPSACK CRYPTOSYSTEM**" is an original work done under the esteemed guidance of **Ms. J. Hyma** and submitted to Department of Computer Science and Engineering, GITAM UNIVERSITY for the partial fulfillment of the requirements for the award of B.Tech Degree. We assure that this project is not submitted in any other university or college.

Regd.No	Name	Signature
1210311234	Mummana Vinod Kumar	
1210311228	Kollipara Vaibhav	
1210311230	Marripati Srikrishna Karteek	
1210311266	Yellapu Bhargava Rajesh	

ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and thanks to our professors and friends in supporting us in accomplishing this project.

Firstly, we would like to thank our project guide **Ms. J. Hyma, Assistant professor** for her unconditional support and guidance which had a greater contribution in completing this project.

Our heartfelt thanks to **Dr. P.V.NAGESWARA RAO, Head of the Department**, **Computer Science and Engineering** for helping us in this project.

We would like to express our special gratitude to our Project Coordinators **Dr. T.Srinivasa Rao, Associate Professor** and **Dr. K.Srinivasa Rao, Associate Professor** for inspiring us, this provided motivation to deeply work on this project and your valuable suggestions are boundless.

Finally, we place a deep sense of gratitude to our staff and our friends who have been constant source of inspiration during the preparation of this project work.

Abstract-

In the recent years, internet multimedia applications have become very popular. Valuable multimedia content such as digital images, however is vulnerable to unauthorized access while in storage and during transmission over a network. As the computers are more and more integrated via the network the distribution of digital media is becoming faster. For effective protection of intellectual properties like security, integrity, confidentiality of the organization's data is an important issue. This project deals with the confidentiality of the data that organization manages and works with. And also proposes a new approach to encrypt the image using the concept of Merkle - Hellman Knapsack Cryptosystem. The Challenge is to design a system that would operate on extremes of Scalability, Usability and Accountability of digital data.

TABLE OF CONTENTS

1. Introduction

Today internet users demand not only text but also audio, images, and video. In order to satisfy user demand need for security and privacy have come to existence. The threat of unauthorized access during transmission over networks and the threat of illegal copy increase significantly. Image encryption, therefore can be used to minimize these problems. Each type of data has its own features, therefore different techniques could be used to protect confidential data from unauthorized access. Hence a novel approach is proposed based on Merkle Hellman cryptosystem. In general message is considered as a plain text. The process of disguising a message in such a way has to hide its substance is encryption. An encrypted message is cipher text. The process of turning cipher text back in to plain text is decryption. There are two types of key-based algorithms one is symmetric and the other one is asymmetric key (public key-private key algorithm). Symmetric key algorithms requires that the sender and receiver agree on a key before they can communicate securely. Complexity theory provides a methodology for analyzing the computational complexity of different cryptographic techniques and algorithms. It compares cryptographic algorithms and techniques and determines their security. An algorithm's complexity is determined by the computational power needed to execute it. The computational complexity is measure by time complexity and space complexity as a function size of the input. There are other measures of complexity: the number of random bits, the communications bandwidth, the amount of data, and so on. The concept of public-key cryptography is its contribution that keys could come in pairs-an encryption key and a decryption key-and that it could be infeasible to generate one key from the other Merkle-Hellman is an asymmetric-key cryptosystem, meaning that for communication. The public key is used only for encryption, and the private key is used only for decryption. The Merkle-Hellman system is based on the sub set problem. In Merkle-Hellman, the keys are knapsacks. The public key is a 'hard' knapsack, and the private key is an 'easy', or super increasing, knapsack, combined with two additional numbers, a multiplier and a modulus, which were used to convert the super increasing knapsack into the hard

knapsack. These same numbers are used to transform the sum of the subset of the hard knapsack into the sum of the subset of the easy knapsack, which is solvable in polynomial time.

1.1 Cryptography

The word cryptography comes from the Greek words κρυπτο (hidden or secret) and γραφη (writing). Oddly enough, cryptography is the art of secret writing. More generally, people think of cryptography as the art of mangling information into apparent unintelligibility in a manner allowing a secret method of unmangling. The basic service provided by cryptography is the ability to send information between participants in a way that prevents others from reading it. In this book we will concentrate on the kind of cryptography that is based on representing information as numbers and mathematically manipulating those numbers. This kind of cryptography can provide other services, such as

• **Integrity checking**—reassuring the recipient of a message that the message has not been altered since it was generated by a legitimate source

• **Authentication**—verifying someone's (or something's) identity But back to the traditional use of cryptography.

A message in its original form is known as **plaintext** or **cleartext**. The mangled information is known as **ciphertext**. The process for producing ciphertext from plaintext is known as **encryption**. The reverse of encryption is called **decryption**.

While cryptographers invent clever secret codes, cryptanalysts attempt to break these codes. These two disciplines constantly try to keep ahead of each other. Cryptographic systems tend to involve both an algorithm and a secret value. The secret value is known as the key. The reason for having a key in addition to an algorithm is that it is difficult to keep devising new algorithms that will allow reversible scrambling of information, and it is difficult to quickly explain a newly devised algorithm to the person with whom you'd like to start communicating securely. With a good cryptographic scheme it is perfectly OK to have everyone, including the bad guys (and the cryptanalysts) know the algorithm because knowledge of the algorithm without the key does not help unmangle the information.

The concept of a key is analogous to the combination for a combination lock. Although the concept of a combination lock is well known (you dial in the secret numbers in the correct sequence and the lock opens), you can't open a combination lock easily without knowing the combination.

1.1.1 Types of Cryptography

There are several ways of classifying cryptographic algorithms. They will be categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The two types of algorithms are:

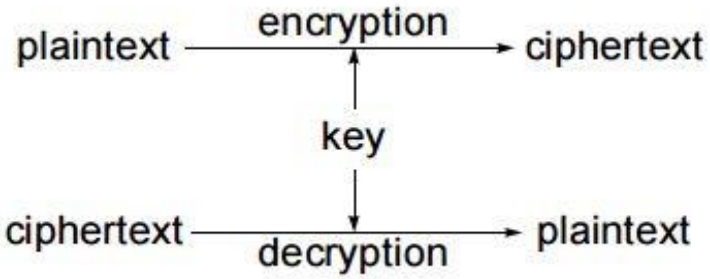
- Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption
- Public Key Cryptography (PKC): Uses one key for encryption and another for decryption

Secret Key Cryptography (SKC)

With *secret key cryptography*, a single key is used for both encryption and decryption. The sender uses the key (or some set of rules) to encrypt the plaintext and sends the ciphertext to the receiver. The receiver applies the same key (or ruleset) to decrypt the message and recover the plaintext.

Because a single key is used for both functions, secret key cryptography is also called *symmetric encryption*.

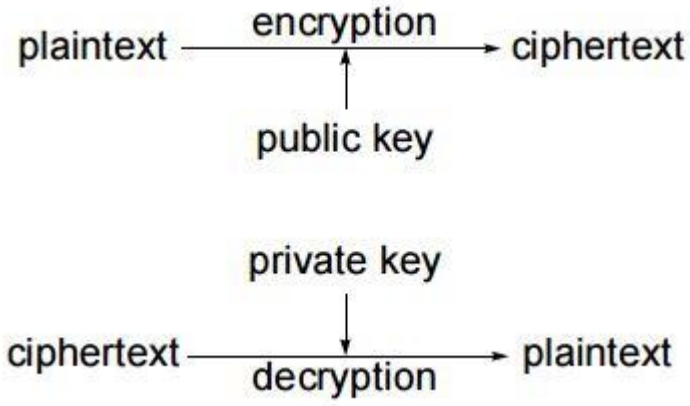
With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.



Secret key cryptography schemes are generally categorized as being either *stream ciphers* or *block ciphers*. Stream ciphers operate on a single bit (byte or computer word) at a time and implement some form of feedback mechanism so that the key is constantly changing. A block cipher is so-called because the scheme encrypts one block of data at a time using the same key on each block. In general, the same plaintext block will always encrypt to the same ciphertext when using the same key in a block cipher whereas the same plaintext will encrypt to different ciphertext in a stream cipher.

Public Key Cryptography (PKC)

Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it does not matter which key is applied first, but that both keys are required for the process to work. Because a pair of keys are required, this approach is also called *asymmetric cryptography*.



In PKC, one of the keys is designated the *public key* and may be advertised as widely as the owner wants. The other key is designated the *private key* and is never revealed to another party. It is straight forward to send messages under this scheme.

1.2 Image Representation

Pixel

Pixel is the smallest element of an image. Each pixel correspond to any one value. In an 8-bit gray scale image, the value of the pixel between 0 and 255. The value of a pixel at any point correspond to the intensity of the light photons striking at that point. Each pixel store a value proportional to the light intensity at that particular location.

Bits Per Pixel (BPP)

Bpp or bits per pixel denotes the number of bits per pixel. The number of different colors in an image is depends on the depth of color or bits per pixel.

If we devise a formula for the calculation of total number of combinations that can be made from bit, it would be like this.

$$(2)^{bpp}$$

Where bpp denotes bits per pixel. Put 1 in the formula you get 2, put 2 in the formula, you get 4. It grows exponentially.

Number of different colors

The number of different colors depend on the number of bits per pixel.

The table for some of the bits and their color is given below.

Bits per pixel	Number of colors
1 bpp	2 colors
2 bpp	4 colors
3 bpp	8 colors
4 bpp	16 colors
5 bpp	32 colors
6 bpp	64 colors
7 bpp	128 colors
8 bpp	256 colors
10 bpp	1024 colors
16 bpp	65536 colors
24 bpp	16777216 colors (16.7 million colors)
32 bpp	4294967296 colors (4294 million colors)

1.2.1 Bit Color Formats:

2, 3, 4, 5, 6 bit color format

The images with a color format of 2, 3, 4, 5 and 6 bit are not widely used today. They were used in old times for old TV displays, or monitor displays.

But each of these colors have more than two gray levels, and hence has gray color unlike the binary image.

In a 2 bit 4, in a 3 bit 8, in a 4 bit 16, in a 5 bit 32, in a 6 bit 64 different colors are present.

8 bit color format

8 bit color format is one of the most famous image format. It has 256 different shades of colors in it. It is commonly known as Grayscale image.

The range of the colors in 8 bit vary from 0-255. Where 0 stands for black, and 255 stands for white, and 127 stands for gray color.

This format was used initially by early models of the operating systems UNIX and the early color Macintoshes.

FORMAT

The format of these images are PGM (Portable Gray Map).

This format is not supported by default from windows. In order to see gray scale image, you need to have an image viewer or image processing toolbox such as Matlab.

BEHIND GRAY SCALE IMAGE

As we have explained it several times in the previous tutorials, that an image is nothing but a two dimensional function, and can be represented by a two dimensional array or matrix. So in the case of the image of Einstein shown above, there would be two dimensional matrix in behind with values ranging between 0 and 255.

But that's not the case with the color images.

16 bit color format

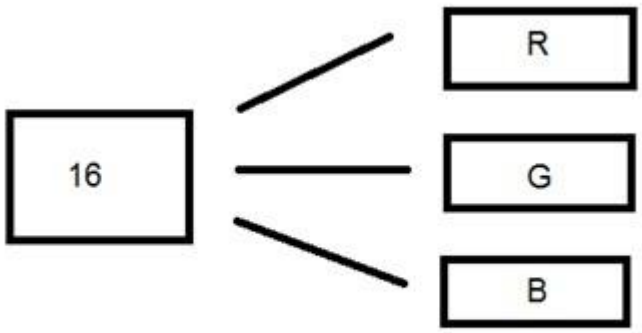
It is a color image format. It has 65,536 different colors in it. It is also known as High color format.

It has been used by Microsoft in their systems that support more than 8 bit color format. Now in this 16 bit format and the next format we are going to discuss which is a 24 bit format are both color format.

The distribution of color in a color image is not as simple as it was in grayscale image.

A 16 bit format is actually divided into three further formats which are Red, Green and Blue. The famous (RGB) format.

It is pictorially represented in the image below.



Now the question arises, that how would you distribute 16 into three. If you do it like this,

5 bits for R, 5 bits for G, 5 bits for B

Then there is one bit remains in the end.

So the distribution of 16 bit has been done like this.

5 bits for R, 6 bits for G, 5 bits for B.

The additional bit that was left behind is added into the green bit. Because green is the color which is most soothing to eyes in all of these three colors.

Note this is distribution is not followed by all the systems. Some have introduced an alpha channel in the 16 bit.

ANOTHER DISTRIBUTION OF 16 BIT FORMAT IS LIKE THIS

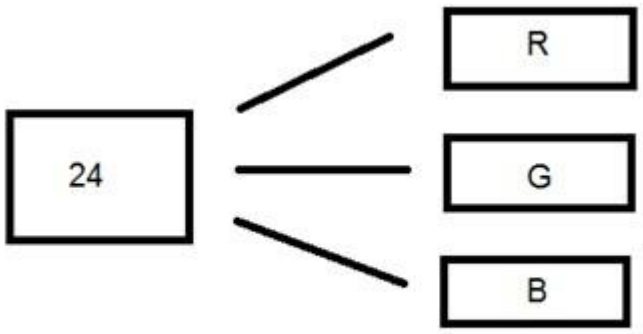
4 bits for R, 4 bits for G, 4 bits for B, 4 bits for alpha channel.

Or some distribute it like this

5 bits for R, 5 bits for G, 5 bits for B, 1 bits for alpha channel.

24 bit color format

24 bit color format also known as true color format. Like 16 bit color format, in a 24 bit color format, the 24 bits are again distributed in three different formats of Red, Green and Blue.



Since 24 is equally divided on 8, so it has been distributed equally between three different color channels.

Their distribution is like this.

8 bits for R, 8 bits for G, 8 bits for B.

BEHIND A 24 BIT IMAGE.

Unlike 8 bit gray scale image, which has one matrix behind it, a 24 bit image has three different matrices of R, G, B.

1.2.2 CONVERSION:

CONVERSION FROM RGB TO HEX CODE

Conversion from Hex to RGB is done through this method:

1. Take a color. E.g.: White = (255, 255, 255).
2. Take the first portion e.g. 255.

3. Divide it by 16. Like this:
4. Take the two numbers below line, the factor, and the remainder. In this case it is 15 & 15 which is FF.
5. Repeat the step 2 for the next two portions.
6. Combine all the hex code into one.

   Answer: #FFFFFF

CONVERSION FROM HEX TO RGB

Conversion from hex code to RGB decimal format is done in this way.

Take a hex number. E.g.: #FFFFFF

Break this number into 3 parts: FF FF FF

Take the first part and separate its components: F F

Convert each of the part separately into binary: (1111) (1111)

Now combine the individual binaries into one: 11111111

Convert this binary into decimal: 255

Now repeat step 2, two more times.

The value comes in the first step is R, second one is G, and the third one belongs to B.

Answer: (255 , 255 , 255)

1.3 Merkle-Hellman Knapsack Cryptosystem

Merkle-Hellman is an asymmetric-key cryptosystem, meaning that two keys are required for communication: a public key and a private key. Furthermore, unlike

RSA, it is one-way: the public key is used only for encryption, and the private key is used only for decryption.

The Merkle-Hellman system is based on the subset sum problem (a special case of the knapsack problem). The problem is as follows: given a set of numbers *A* and a number **b**, find a subset of *A* which sums to b. In general, this problem is known to be NP-complete. However, if the set of numbers (called the knapsack) is superincreasing, meaning that each element of the set is greater than the sum of all the numbers before it.

1.3.1 Key generation

In Merkle-Hellman, the keys are two knapsacks. The public key is a 'hard' knapsack *A*, and the private key is an 'easy', or superincreasing, knapsack *B*, combined with two additional numbers, a multiplier and a modulus. The multiplier and modulus can be used to convert the superincreasing knapsack into the hard knapsack. These same numbers are used to transform the sum of the subset of the hard knapsack into the sum of the subset of the easy knapsack.

1.3.2 Encryption

To encrypt a message, a subset of the hard knapsack *A* is chosen by comparing it with a set of bits (the plaintext) equal in length to the key. Each term in the public key that corresponds to a 1 in the plaintext is an element of the subset *A_m*, while terms that corresponding to 0 in the plaintext are ignored when constructing *A_m* - they are not elements of the key. The elements of this subset are added together and the resulting sum is the ciphertext.

1.3.3 Decryption

Decryption is possible because the multiplier and modulus used to transform the easy knapsack into the public key can also be used to transform the number representing the ciphertext into the sum of the corresponding elements of the superincreasing knapsack.

1.3.4 Mathematical Model

Key generation

To encrypt *n*-bit messages, choose a superincreasing sequence

$w = (w_1, w_2, ..., w_n)$

of *n* nonzero natural numbers. Pick a random integer *q*, such that

$$q > \sum_{i=1}^{n} w_i$$

,

and a random integer, *r*, such that gcd(*r*,*q*) = 1 (i.e. *r* and *q* are coprime).

q is chosen this way to ensure the uniqueness of the ciphertext. If it is any smaller, more than one plaintext may encrypt to the same ciphertext. Since *q* is larger than the sum of every subset of *w*, no sums are congruent mod *q* and therefore none of the private key's sums will be equal. *r* must be coprime to *q* or else it will not have an inverse mod *q*. The existence of the inverse of *r* is necessary so that decryption is possible.

Now calculate the sequence

$\beta = (\beta_1, \beta_2, ..., \beta_n)$

where

$\beta_i = rw_i \bmod q$.

The public key is β, while the private key is (*w*, *q*, *r*).

Encryption

To encrypt an *n*-bit message

$\alpha = (\alpha_1, \alpha_2, ..., \alpha_n)$,

where α_i is the *i*-th bit of the message and $\alpha_i \in \{0, 1\}$, calculate

$$c = \sum_{i=1}^{n} \alpha_i \beta_i.$$

The cryptogram then is *c*.

Decryption

In order to decrypt a ciphertext *c* a receiver has to find the message bits α_i such that they satisfy

$$c = \sum_{i=1}^{n} \alpha_i \beta_i.$$

This would be a hard problem if the β_i were random values because the receiver would have to solve an instance of the subset sum problem, which is known to be NP-hard. However, the values β_i were chosen such that decryption is easy if the private key (*w*, *q*, *r*) is known.

The key to decryption is to find an integer *s* that is the modular inverse of *r* modulo *q*. That means *s* satisfies the equation *s r* mod *q* = 1 or equivalently there exist an integer *k* such that *sr* = *kq* + 1. Since *r* was chosen such that gcd(*r*,*q*)=1 it is possible to find *s* and *k* by using the Extended Euclidean algorithm. Next the receiver of the ciphertext computes

$$c' \equiv cs \pmod{q}.$$

Hence

$$c' \equiv cs \equiv \sum_{i=1}^{n} \alpha_i \beta_i s \pmod{q}.$$

Because of *rs* mod q = 1 and $\beta_i = rw_i$ mod *q* follows

$$\beta_i s \equiv w_i r s \equiv w_i \pmod{q}.$$

Hence

$$c' \equiv \sum_{i=1}^{n} \alpha_i w_i \pmod{q}.$$

The sum of all values w_i is smaller than *q* and hence $\sum_{i=1}^{n} \alpha_i w_i$ is also in the interval [0,*q*-1]. Thus the receiver has to solve the subset sum problem

$$c' = \sum_{i=1}^{n} \alpha_i w_i.$$

This problem is easy because *w* is a superincreasing sequence. Take the largest element in *w*, say w_k. If $w_k > c'$, then $\alpha_k = 0$, if $w_k \leq c'$, then $\alpha_k = 1$. Then, subtract $w_k \times \alpha_k$ from c' , and repeat these steps until you have figured out α.

2. Review Literature

2.1 JAVA

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and the Mike Sheridan at Sun Microsystems Inc.in 1991.It took 18 months to develop the first working version. This language was initially called 'Oak' but was renamed as "Java" in 1995.Between the initial implementation of Oak in the fall of 1992 and the public announcement of Java in the spring of 1996,many more people contributed to the design and evolution of the language.
The main properties of the Java, which made Java so popular, are as follows

- Simple
- Secure
- Portable
- Object Oriented
- Robust
- Multithreaded
- Architecture-Neutral
- Integrated
- High Performance
- Distributed
- Dynamic

The most striking feature of the language is that is a platform neutral language. Java is first programming language that is not tied to any particular hardware or operating system. Programs developed in Java can be executed anywhere on any system.

The Key Features Of Java Is Byte Code

The key that allows Java to solve both the security and the portability problems just described is that the output of a Java compiler is not executable code. Rather, it is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java runtime systems, which is called the Java Virtual Machine (JVM).That is, in its standard form, the JVM is an interpreter for Byte code. This may come has a bit of surprise.

Translating a Java program into a byte code helps and makes it much easier to run a program in a wide variety of environments. The reason is straightforward only the JVM needs to be implemented for each platform. Once the runtime package exists for a given system, any Java program can run on it. Remember, although the details of the JVM will differ from platform to platform, all interpret the same Java Byte code.

Java Environment

Java environment includes a large number of development tools and hundreds of classes and methods. The development tools are the part of the system known as Java Development Kit (JDK) and the classes are methods are part of the Java standard library (JSL),also known as the Application Programming Interface.

Java Development Kit

The Java development kit comes with a collection of tools that are used for developing and running Java programs .They include:

1. Applet Viewer (for viewing Java Applets)
2. Javac (Java compiler)
3. Java (Java Interpreter)
4. Javap (Java Disassembler)
5. Javadoc (for creating HTML documents)
6. Javah (for C header files)
7. Jdb (Java Debugger)

Application Programming Interface

The Java standard library includes hundreds of classes and methods grouped into several functional packages .Most commonly used packages are:

Ø Language support package: A collection of classes and methods required for implementing basic features of java.

Ø Utilities packages: A collection of classes to provide utility functions such as date and time functions.

Ø Input/Output: A collection of classes required for input & output manipulations.

Ø Networking package: A collection of classes for communication with other computers via internet.

Ø AWT package: The abstract window toolkit package contains classes that implements platform independent graphical user interface.

Ø Applet package: This includes a set of classes that allows us to create Java applets.

HTML:

The Hypertext Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another.HTML documents are SGML documents with the generic Semantics that are appropriate for representing the information from wide range of applications.Html version 3.2

A set of instruction embedded in a document is called the markup language. These instructions describe what the document text means and how it should look like in a display hyperlink specification language that define the syntax and placement of special embedded directories that are not displayed by a web browser, but it tell how to displays the content of the documents including text, images and other supported media.

Web site is a collection of pages, publications and documents that reside on the web server. While these page publications and a document as a formatted in any single format. You should use html for Home page and all primary pages and the site. This will enable the millions of web users to easily access and to take advantage of your website.Html Documents are platform independent, if they created properly you can move home page to any server platform.

Basic Markup Tags

An HTML documents starts with <HTML> tag. This element tells the browser that the file contains HTML coded information.HTML document contain two parts namely the head and body.

Head

<HEAD>…</HEAD>

This is the first part of HTML document, which contain description of the HTML page.

Title

<TITLE>…</TITLE>

These tags are embedded within the Head tags. Each HTML page should have a short and descriptive title. The browser at the top usually displays the title.

Ø Is used in the indexes as well as in browser's history list and bookmarks.

Ø Cannot contain any formatting, images, or links to other pages.

Ø Can have animated titles.

Example

< HTML>

<HEAD>

<TITLE>this is an html title</TITLE>

</HEAD>

</HTML>

Advantages

An HTML document is small and hence easy to send over the net. It is small because it does not include format information.

HTML document are cross platform compatible and device independent. We only need an HTML readable browser to view them. For names, locations etc. are not required.

JAVA SCRIPT

Java script is a scripting language developed jointly by sun Netscape and is meant for the WWW. A scripting language is a simple script based programming language designed to enable programming to write useful programs quickly. A script is similar to a macro, which tells a program how to perform a specific procedure. As you go through this chapter we will get a better understand of what a scripting is how Java Script brings about interactive web pages with HTML.

Java Script is embedded into a Html

Java Script code usually embedded into HTML document and is executed within them. By itself JavaScript has no user interface. It rallies on HTML to provide the mean of interaction with the user. Most of JavaScript object have

HTML by providing events to HTML tags and provide event driven code to execute it.

Java Script is Browser Dependent

JavaScript depends on the web browser to support it. If the browser doesn't support it, JavaScript code will be ignored. Internet Explorer 3.0 and Netscape Navigator 2.0 onwards support JavaScript.

Java Script is An Interpreted Language

Java script is interpreted at runtime by the browser before it is executed. It is not completed into a separate program like a exe but remains part of the HTML file.

Java Script Is A Loosely Type Language

Java script is very flexible compared to java. We need not specify the data type of a variable while declaring it. Also we need not declare variable explicitly. It is perfectly legal to declare variable as when we required them.

Java Script Is an Object-Based Language

Java Script is an object-based language. We can work with objects that encapsulate data and behavior. However JavaScript object model is instanced-based and there is no inheritance. This is basic difference between an object oriented and objects based language.

JavaScript is Not Java

Java applet is stored in a separate file and connected to HTML file through the <applet> tag and it strongly typed, object oriented compiled language. JavaScript is loosely typed object based, interpreted language meant to create script JavaScript can be used to

- Ø Enhance Html pages
- Ø Develop client side application
- Ø Built to a certain extend client/server web application.
- Ø Create extension to a web server.
- Ø Provide database connectivity without using CGI.

Client Side Framework

The client side framework include the following

Web Browser

Html client extension scripts language JavaScript role in web application development.

Client Side Application

JavaScript has good capabilities when working with Html tags & associated objects compared to java. For certain cases JavaScript provides a programming backbone with which to develop application.

Data Validation

JavaScript provides the means for basic data validation before it is sent to the server. Whether the values entered are correct or not or whether all the fields in a form are filled out or not can be checked before sending data to web server, if JavaScript is not used then data is sent to web server and the web server would response with a message that the data sent to it is incorrect or incomplete. Thus JavaScript ensures data validation and also reduces the network traffic.

Java Script Object Model

JavaScript is an object-based language. It has no inheritance. The relationship between objects at different levels is not ancestor descent but of container. When an object properties or methods are referenced is used to denote ownership.

Document Object

Document object is the most important as the document object is responsible for all the actual contents displayed on a given page. Using document object we can display dynamic Html pages. Also all typical interface elements of a web application are contained in the document. A common use of document object is generating Html pages through JavaScript. This is done using write() or writeln() methods.

JAVA SERVER PAGE(JSP)

Java Server Pages is a simple, yet powerful technology for creating and maintaining dynamic-content web pages. Based on the Java programming language, Java Server Pages offers proven portability, open standards and a mature re-usable component model.

Portability

Java Server Pages files can be run on any web server or web enabled application server that provides support for them. Dubbed the JSP engine, this

support involves recognition, translation and management of the Java Server Pages lifecycle and its interaction with associated components.

The JSP engine for a particular server might be built-in or might be provided through a 3rd –party add-on. As long as the server on which you plan to execute the Java Server Pages supports the same specification level as that to which the file was written, no change should be necessary as you move your files from server to server. Note, however, that instructions for the setup and configuration of the files may differ between files.

Composition

It was mentioned earlier that the Java Server Pages architecture could include reusable Java components. The architecture also allows for the embedding of a scripting language directly into the Java Server Pages file. The components current supported include Java Beans and Servers. As the default scripting language, Java Server Pages use the Java Programming language. This means that scripting on the server side can take advantage of the full set of capabilities that the Java programming language offers.

Processing

A Java Server Pages file is essentially an HTML document with JSP scripting or tags. It may have associated components in the form of .class, .jar, or .ser files –or it may not. The use of components is not required.

The Java Server Pages file has a .jsp extension to identify it to the server as a Java Server Pages file. Before the page is served, the Java Server Pages syntax is parsed and processed into a servlet on the server side. The servlet that is generated, outputs real content in straight HTML for responding to the customer. Because it is standard HTML, the dynamically generated response looks no different to the customer browser that a static response.

SERVLET

A servlet is a small program that runs on a server. A servlet acts as an intermediary between the client and the server. As servlet modules run on the server, they can receive and respond to requests made by the client. Request and

response objects of the servlet offer a convenient way to handle HTTP requests and send text data back to the client.

Since a servlet is integrated with the Java language, it also possesses all the Java features such as high portability, platform independence, security and Java database connectivity.

There are two Java Servlet types: Basic and HTTP.

HTTP servlets are used as follows:

- When an HTML form is submitted, the servlet processes and stores the data.
- When a client supplies a database query, the results are provided to the client by the servlet
- In most cases, the server uses the common gateway interface (CGI).

However, Java Servlets have many advantages over CGI, including:

- A servlet runs in the same process, eliminating the need to create a new process for every request.
- The CGI program must be reloaded for each CGI request. A servlet, however, does not require reloading and remains in the memory between requests.
- A servlet answers multiple requests simultaneously by using one instance, saving memory and easily managing persistent data.
- The servlet engine runs in a sandbox or restricted environment, protecting the server from potentially harmful servlets

2.2 Java EE

Java Platform, Enterprise Edition or **Java EE** is Oracle's enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications. Java EE extends the Java Platform, Standard Edition (Java SE), providing an API for object relational mapping, distributed and multi-tier architectures, and web services. The platform incorporates a design based largely on modular components running on an application server. Software for Java EE is primarily developed in the Java programming language. The platform emphasizes

Convention over configuration and annotations for configuration. Optionally XML can be used to override annotations or to deviate from the platform defaults.

Version History

The platform was known as Java 2 Platform, Enterprise Edition or J2EE until the name was changed to Java Platform, Enterprise Edition or Java EE in version 5. The current version is called Java EE 7.

- J2EE 1.2 (December 12, 1999)
- J2EE 1.3 (September 24, 2001)
- J2EE 1.4 (November 11, 2003)
- Java EE 5 (May 11, 2006)
- Java EE 6 (Dec 10, 2009)
- Java EE 7 (May 28, 2013), but April 5, 2013 according to spec document.

Standards and specifications

Java EE is defined by its specification. As with other Java Community Process specifications, providers must meet certain conformance requirements in order to declare their products as Java EE compliant.

Java EE includes several API specifications, such as JDBC, RMI, e-mail, JMS, web services, XML, etc., and defines how to coordinate them. Java EE also features some specifications unique to Java EE for components. These include Enterprise JavaBeans, Connectors, servlets, JavaServer Pages and several web service technologies. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. A Java EE application server can handle transactions, security, scalability, concurrency and management of the components it is deploying, in order to enable developers to concentrate more on the business logic of the components rather than on infrastructure and integration tasks.

General API's

The Java EE APIs includes several technologies that extend the functionality of the base Java SE APIs.

- Java EE 7 Platform Packages
- Java EE 6 Platform Packages

- Java EE 5 Platform Packages

javax.servlet.*;

The servlet specification defines a set of APIs to service mainly HTTP requests. It includes the JavaServer Pages (JSP) specification.

javax.websocket.*;

The Java API for WebSocket specification defines a set of APIs to service WebSocket connections.

javax.faces.*;

This package defines the root of the JavaServer Faces (JSF) API. JSF is a technology for constructing user interfaces out of components.

javax.faces.component.*;

This package defines the component part of the JavaServer Faces (JSF) API. Since JSF is primarily component oriented, this is one of the core packages. The package overview contains a UML diagram of the component hierarchy.

2.3 Apache Tomcat Server

Apache Tomcat (or simply **Tomcat**, formerly also *Jakarta Tomcat*) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run in. In the simplest configuration Tomcat runs in a single operating system process. The process runs a Java virtual machine (JVM). Every single HTTP request from a browser to Tomcat is processed in the Tomcat process in a separate thread. Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

Components:

Tomcat 4.x was released with Catalina (servlet container), Coyote (a HTTP connector) and Jasper (a JSP engine).

Catalina

Catalina is Tomcat's servlet container. Catalina implements Sun Microsystems' specifications for servlet and JavaServer Pages (JSP). In Tomcat, a Realm element represents a "database" of usernames, passwords, and roles (similar to Unix groups) assigned to those users. Different implementations of Realm allow Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification.

Coyote

Coyote is Tomcat's HTTP Connector component that supports the HTTP 1.1 protocol for the web server or application container. Coyote listens for incoming connections on a specific TCP port on the server and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client. It can execute JSP's and Servlets.

Jasper

Jasper is Tomcat's JSP Engine. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.

As of version 5, Tomcat uses Jasper 2, which is an implementation of the Sun Microsystems's JSP 2.0 specification. From Jasper to Jasper 2, important features were added:

- JSP Tag library pooling - Each tag markup in JSP file is handled by a tag handler class. Tag handler class objects can be pooled and reused in the whole JSP servlet.
- Background JSP compilation - While recompiling modified JSP Java code, the older version is still available for server requests. The older JSP servlet is deleted once the new JSP servlet has finished being recompiled.
- Recompile JSP when included page changes - Pages can be inserted and included into a JSP at runtime. The JSP will not only be recompiled with JSP file changes but also with included page changes.
- JDT Java compiler - Jasper 2 can use the Eclipse JDT (Java Development Tools) Java compiler instead of Ant and javac.

Three new components were added with the release of Tomcat 7:

Cluster

This component has been added to manage large applications. It is used for load balancing that can be achieved through many techniques. Clustering support currently requires the JDK version 1.5 or later.

High availability

A high-availability feature has been added to facilitate the scheduling of system upgrades (e.g. new releases, change requests) without affecting the live environment. This is done by dispatching live traffic requests to a temporary server on a different port while the main server is upgraded on the main port. It is very useful in handling user requests on high-traffic web applications.

Web Application

It has also added user as well as system based web applications enhancement to add support for deployment across the variety of environments. It also tries to manage session as well as applications across the network.

Tomcat is building additional components. A number of additional components may be used with Apache Tomcat. These components may be built by users should they need them or they can be downloaded from one of the mirrors.

Features

Tomcat 7.x implements the Servlet 3.0 and JSP 2.2 specifications. It requires Java version 1.6, although previous versions have run on Java 1.1 through 1.5. Versions 5 through 6 saw improvements in garbage collection, JSP parsing, performance and scalability. Native wrappers, known as "Tomcat Native", are available for Microsoft Windows and Unix for platform integration.

2.4 Oracle database

A relational database management system (DBMS) from Oracle, which runs on more than 80 platforms. Introduced in the late 1970s, Oracle was the first database product to run on a variety of platforms from micro to mainframe. The Oracle database is Oracle's flagship product, and version 11g was introduced in 2007.

Oracle 11g features include built-in testing for changes, the capability of viewing tables back in time, superior compression of all types of data and enhanced disaster recovery functions.

The "i" and "g" Versions

Starting in 1999 with Version 8i, Oracle added the "i" to the version name to reflect support for the Internet with its built-in Java Virtual Machine (JVM). Oracle 9i added more support for XML in 2001. In 2003, Oracle 10g was introduced with emphasis on the "g" for grid computing, which enables clusters of low-cost, industry standard servers to be treated as a single unit.

Java Built In

With a JVM (Java interpreter) built into the DBMS, triggers and stored procedures can be written and executed in Java rather than Oracle's PL/SQL programming language. It enables Internet developers to write applications and database procedures in the same language. In addition, the JVM can also execute Enterprise JavaBeans (EJBs), turning the DBMS into an application server.

Physical and logical structures

An Oracle database system—identified by an alphanumeric system identifier or SID—comprises at least one instance of the application, along with data storage. An instance—identified persistently by an instantiation number (or activation id: SYS.V_$DATABASE.ACTIVATION#)—comprises a set of operating-system processes and memory-structures that interact with the storage. (Typical processes include PMON (the process monitor) and SMON (the system monitor).) Oracle documentation can refer to an active database instance as a "shared memory realm".

Users of Oracle databases refer to the server-side memory-structure as the SGA (System Global Area). The SGA typically holds cache information such as data-buffers, SQL commands, and user information. In addition to storage, the database consists of online redo logs (or logs), which hold transactional history. Processes can in turn archive the online redo logs into archive logs (offline redo

logs), which provide the basis (if necessary) for data recovery and for the physical-standby forms of data replication using Oracle Data Guard.

If the Oracle database administrator has implemented Oracle RAC (Real Application Clusters), then multiple instances, usually on different servers, attach to a central storage array. This scenario offers advantages such as better performance, scalability and redundancy. However, support becomes more complex, and many sites do not use RAC. In version 10g, grid computing introduced shared resources where an instance can use (for example) CPU resources from another node (computer) in the grid.

The Oracle DBMS can store and execute stored procedures and functions within itself. PL/SQL (Oracle Corporation's proprietary procedural extension to SQL), or the object-oriented language Java can invoke such code objects and/or provide the programming structures for writing them.

Storage

The Oracle RDBMS stores data logically in the form of tablespaces and physically in the form of data files ("datafiles"). Tablespaces can contain various types of memory segments, such as Data Segments, Index Segments, etc. Segments in turn comprise one or more extents. Extents comprise groups of contiguous data blocks. Data blocks form the basic units of data storage.

A DBA can impose maximum quotas on storage per user within each tablespace.

Partitioning

Newer versions of the database can also include a partitioning feature: this allows the partitioning of tables based on different set of keys. Specific partitions can then be easily added or dropped to help manage large data sets.

Monitoring

Oracle database management tracks its computer data storage with the help of information stored in the SYSTEM tablespace. The SYSTEM tablespace contains the data dictionary—and often (by default) indexes and clusters. A data dictionary consists of a special collection of tables that contains information about all user-objects in the database. Since version 8i, the Oracle RDBMS also supports "locally managed" tablespaces that store space management information in bitmaps in their own headers rather than in the SYSTEM tablespace (as happens with the

default "dictionary-managed" tablespaces). Version 10g and later introduced the SYSAUX tablespace, which contains some of the tables formerly stored in the SYSTEM tablespace, along with objects for other tools such as OEM, which previously required its own tablespace.

3. SYSTEM ANALYSIS

3.1 Problem Statement:

Protecting some of the images are as important as protecting the textual data and perhaps much more important. Image encryption needs to be secure by resisting statistical attacks and other types of attacks.

3.2 Project Objective:

The objective is to develop an application that can be used to encrypt an image file. This application must be simple easy to use, and powerful. Many factors have been considered in order to develop this application such as processing speed of image, the strength of the encryption result and ease of use to end user.

3.3 Existing System:

In the last few years, network security and data encryption have become an important and high profile issues. Innovation encryption techniques need to be developed for effective data encryption for financial institutions, e-commerce, and multimedia applications. For future internet applications on wireless networks, besides source coding and channel coding techniques, cryptographic coding techniques for multimedia applications need to be studied and developed.

3.4 Problem Identification and Problem Definition:

From the study, we analyzed that Images are different from text. Although user may use the traditional cryptosystems to encrypt images directly, it is not a right idea for two reasons. One is that the size of image is almost always much higher than that of text. Therefore, the traditional cryptosystems need too much time to directly encrypt the image data. Another problem is that the decrypted text must be equal to the original or plain text.

3.5 Proposed System:

In this approach each pixel of the image is encrypted considering the super increasing sequence of 8 bits of Merkle Hellman Knapsack cryptosystem this

strengthens the confidentiality of the data, which is the prime necessity of any organization looking forward for data security.

4. REQUIREMENTS AND SPECIFICATIONS

4.1 Software Interface

- **Client on Internet**
  Web Browser, Operating System

- **Web Server**
  Apache Tomcat Server

- **Data Base Server**
  Oracle 11g

- **Development End**
  J2EE, Java, Servlets, HTML, XML, CSS, OS (Windows), Apache Tomcat Server (Web Server)

4.2 Hardware Interface

- **Client Side**
  Web Browser
  Processor -> Intel Pentium III or AMD - 800 MHz
  RAM -> 128
  Disk Space -> MB 100 MB
- **Server Side**
  Processor -> Intel Pentium III or AMD - 800 MHz
  RAM -> 1GB
  Disk Space -> 3.5GB
- **Oracle 11g**
  Processor -> Intel Pentium III or AMD - 800 MHz
  Disk Space -> 256 MB
  Disk Space -> 500 MB (Excluding Data Size)

4.3 Communication Interface

- Client (customer) on Internet will be using HTTP/HTTPS protocol.

- Client (system user) on Internet will be using HTTP/HTTPS protocol.

4.4 Constraints

- GUI is only in English.
- Login and password is used for the identification of users.
- Only registered admins and users will be authorized to use the services. Limited to HTTP/HTTPS.

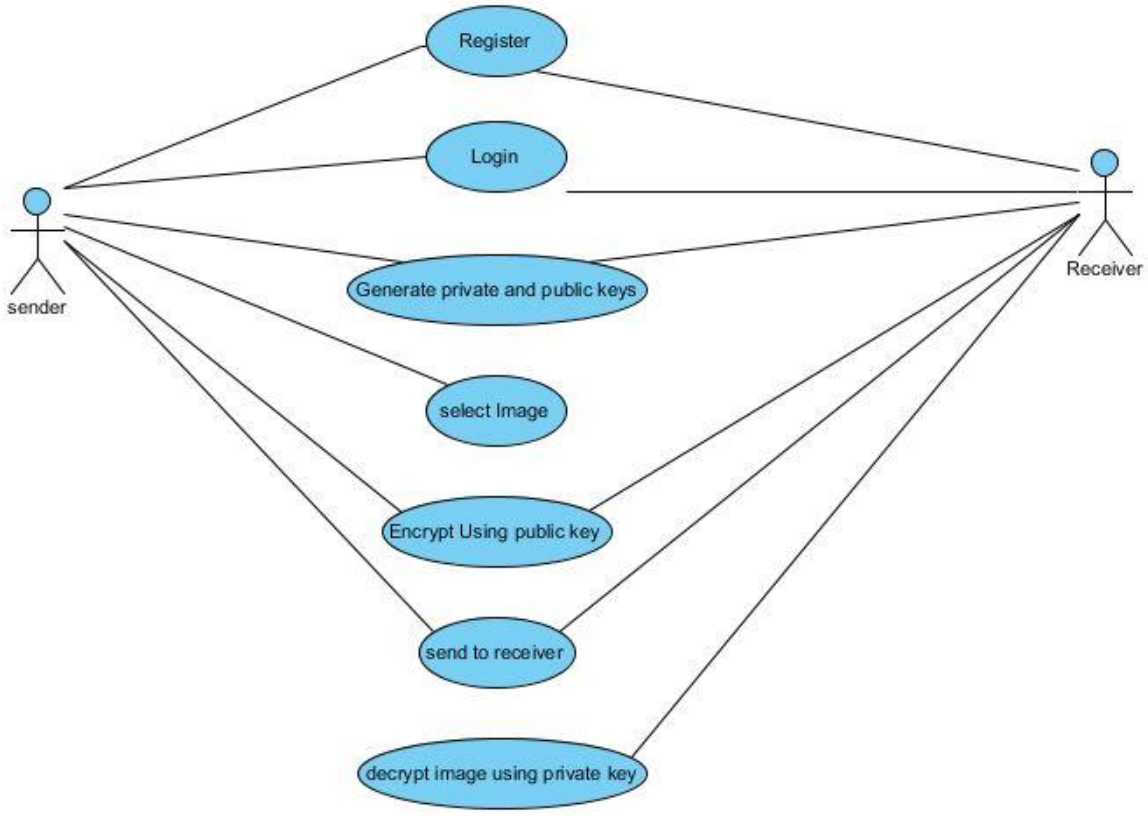
5. SYSTEM MODELING

5.1 Unified Modeling Language (UML) Diagrams

Diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). In theory, a diagram may contain any combination of things and relationships.

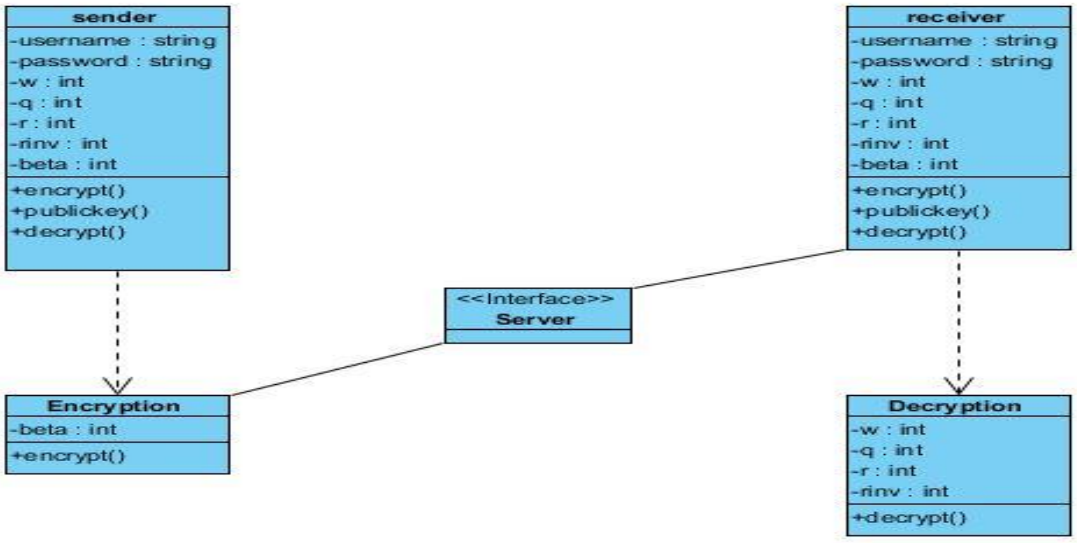
5.1.1 USE CASE Modeling: Use Case Diagram:

A use case diagram shows a set of use cases and actors and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system.



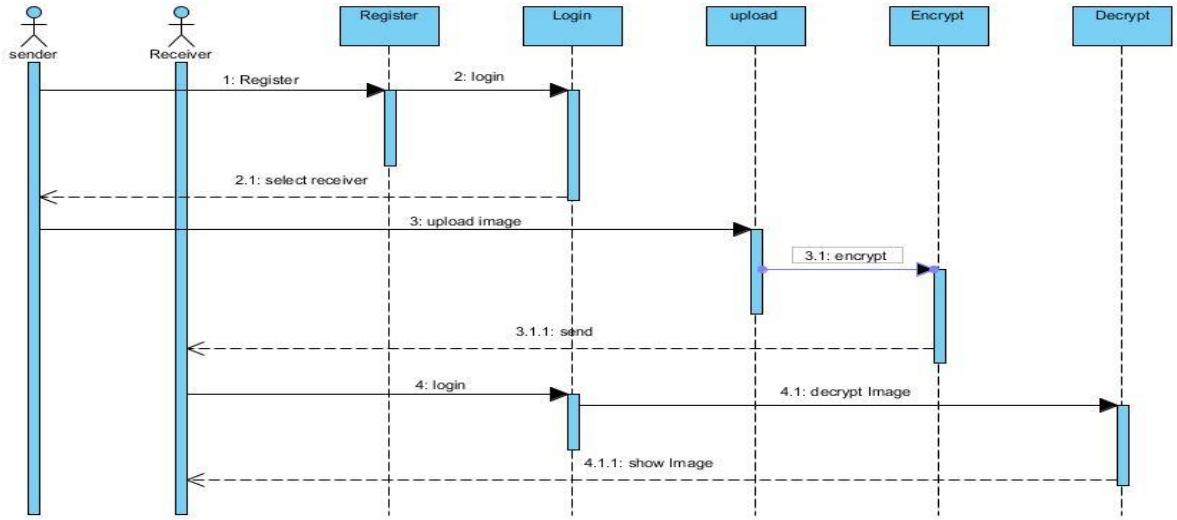
5.1.2 Structural Modeling: Class Diagram

A class diagram shows a set of classes, interfaces, and collaborations and their relationships. Class diagrams that include active classes address the static process view of a system.



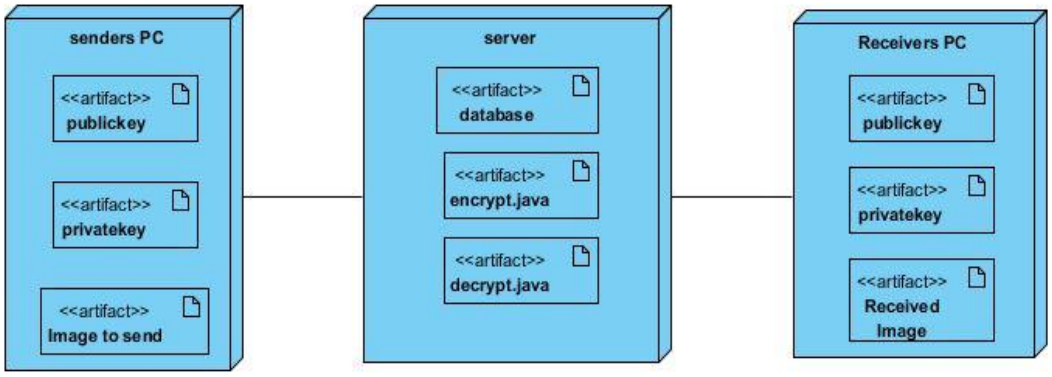
5.1.3 Behavioral Modeling: Sequence Diagram

A sequence diagram is an interaction diagram that emphasizes the time ordering of messages.



5.1.4 Architectural Modeling: Deployment Diagram

A deployment diagram shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams address the static deployment view of an architecture



6. IMPLEMENTATION

SOURCE CODE:

Main Algorithm:

ImageEncryption.java

```
package com.merhell;

import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.util.Random;

import javax.imageio.ImageIO;

public class ImgEnc {

	private int w[];
	private int q, r, rinv;
	public int beta[];

	public int[] privatekey() {
		int sum, i = 0;
		w = new int[10];
		Random rand = new Random();
		sum = 0;
		while (i < 8) {
			w[i] = rand.nextInt(sum + 3);
			if (w[i] > sum) {
				sum += w[i++];
				System.out.print(w[i - 1] + " ");
			}
		}
		System.out.println("\nSum:(sigma w)=" + sum);
		// q and r generation
		q = r = 0;
		while (q < sum)
```

```
                q = rand.nextInt(1000);
        // checking for co-prime
        while (r == 0) {
                r = rand.nextInt(q);
                for (i = 2; i <= r && (r % i != 0 || q % i != 0); i++)
                        ;
                if (i > r)
                        break;
                else
                        r = 0;
        }
        System.out.println("q=" + q + "  r=" + r);
        w[8] = q;
        w[9] = r;
        return w;
}

public int[] publickey() {
        beta = new int[10];
        for (int i = 0; i < 8; i++) {
                beta[i] = (w[i] * r) % q;
                System.out.print(beta[i] + "  ");
        }
        return beta;
}

public BufferedImage[] encryption(BufferedImage image, int[] beta) {

        try {
                int width = image.getWidth();
                int height = image.getHeight();
                BufferedImage img[] = new BufferedImage[2];
                img[0] = new BufferedImage(width, height,
                                BufferedImage.TYPE_INT_ARGB);
                img[1] = new BufferedImage(width, height,
                                BufferedImage.TYPE_INT_RGB);
                Color c_original, c_duplicate, c_code;
                for (int i = 0; i < height; i++) {
                        for (int j = 0; j < width; j++) {
                                c_original = new Color(image.getRGB(j, i));
                                int red = c_original.getRed();
                                int green = c_original.getGreen();
                                int blue = c_original.getBlue();
                                String redstring = "";
```

```
                                        String greenstring = "";
                                        String bluestring = "";
                                        for (int k = 0; k < 8; k++) {
                                                redstring += red % 2;
                                                red = red / 2;
                                                greenstring += green % 2;
                                                green = green / 2;
                                                bluestring += blue % 2;
                                                blue = blue / 2;
                                        }
                                        redstring = new StringBuffer(redstring).reverse()
                                                        .toString();
                                        greenstring = new StringBuffer(greenstring).reverse()
                                                        .toString();
                                        bluestring = new StringBuffer(bluestring).reverse()
                                                        .toString();
                                        int ered = 0, egreen = 0, eblue = 0;
                                        int n = 8;
                                        while (--n >= 0) {
                                                if (redstring.charAt(n) == '1')
                                                        ered += beta[n];
                                                if (greenstring.charAt(n) == '1')
                                                        egreen += beta[n];
                                                if (bluestring.charAt(n) == '1')
                                                        eblue += beta[n];
                                        }
                                        c_code = new Color(Math.abs(ered / 256),
                                                        Math.abs(egreen / 256), Math.abs(eblue /
256),
                                                        Math.abs(5));
                                        c_duplicate = new Color(Math.abs((ered) % 256),
                                                        Math.abs((egreen) % 256),
Math.abs((eblue) % 256));
                                        img[0].setRGB(j, i, c_code.getRGB());
                                        img[1].setRGB(j, i, c_duplicate.getRGB());
                                }
                        }
                        return img;
                } catch (Exception e) {
                        e.printStackTrace();
                        System.out.println("Problem !!");
                }
                return null;
        }
```

```
	public BufferedImage decryption(int p[],String fname) {
		BufferedImage image, code;
		int width;
		int height;
		BufferedImage img;
		Color c_original, c_duplicate, c_code;
		q = p[8];
		r = p[9];
		Random rand = new Random();
		rinv = rand.nextInt(1000);
		System.out.println("q : "+q+"\nr : "+r+"\nrinv : "+rinv);
		while ((r * rinv) % q != 1 && rinv !=0)
			rinv = rand.nextInt(1000);
		try {
			File input = new File(
					"E:\\Java EE
Eclipse\\MyProjects\\FinalProject\\encrypted_images\\image_"
							+ fname + ".png");
			image = ImageIO.read(input);
			width = image.getWidth();
			height = image.getHeight();
			System.out.println("width :" + width + "\nheight :" + height);
			img = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
			File image1 = new File(
					"E:\\Java EE
Eclipse\\MyProjects\\FinalProject\\encrypted_images\\code_"
							+ fname + ".png");
			code = ImageIO.read(image1);
			for (int i = 0; i < height; i++) {
				for (int j = 0; j < width; j++) {
					c_original = new Color(image.getRGB(j, i));
					c_code = new Color(code.getRGB(j, i));
					int red = c_original.getRed();
					red = red + 256 * c_code.getRed();
					int green = c_original.getGreen();
					green = green + 256 * c_code.getGreen();
					int blue = c_original.getBlue();
					blue = blue + 256 * c_code.getBlue();
					int dred = (red * rinv) % q;
					int dgreen = (green * rinv) % q;
					int dblue = (blue * rinv) % q;
					String x = "00000000";
					char[] xChars = x.toCharArray();
```

```
                                String y = "00000000";
                                char[] yChars = y.toCharArray();
                                String z = "00000000";
                                char[] zChars = z.toCharArray();
                                xChars = check(dred, xChars, p);
                                x = String.valueOf(xChars);
                                yChars = check(dgreen, yChars, p);
                                y = String.valueOf(yChars);
                                zChars = check(dblue, zChars, p);
                                z = String.valueOf(zChars);
                                x = new StringBuffer(x).reverse().toString();
                                y = new StringBuffer(y).reverse().toString();
                                z = new StringBuffer(z).reverse().toString();
                                int count = 7, xd, yd, zd;
                                xd = yd = zd = 0;
                                while (count >= 0) {
                                        if (x.charAt(count) == '1')
                                                xd = (int) (xd + Math.pow(2, count));
                                        if (y.charAt(count) == '1')
                                                yd = (int) (yd + Math.pow(2, count));
                                        if (z.charAt(count) == '1')
                                                zd = (int) (zd + Math.pow(2, count));
                                        count--;
                                }
                                c_duplicate = new Color(Math.abs(xd), Math.abs(yd),
                                                Math.abs(zd));
                                img.setRGB(j, i, c_duplicate.getRGB());
                        }
                }
                System.out.println("Image Created");
                return img;
        } catch (Exception e) {
                e.printStackTrace();
                System.out.println("Problem !!");
                return null;
        }
}

public char[] check(int dec, char[] Chars, int[] p) {
        while (dec > 0) {
                int count = 0;
                while (dec >= p[count] && count < 8) {
                        count++;
                }
```

```
                        count--;
                        dec = dec - p[count];
                        Chars[count] = '1';
                }
                return Chars;
        }
}
```

Home Page:

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merkle Hellman Knapsack Cryptosystem</title>
<link rel="stylesheet" type="text/css" href="mystyle.css"/>
</head>
<body>

        <div id="header">
                <h1>Image Encryption Using Merkle-Hellman Knapsack Cryptosystem</h1>
  </div>

<!-- ____________________________________________________________________ -->

  <div id="main">
   <div class="box">
     <h2>
       <a href="login.jsp">Login</a>
     </h2>
   </div>
   <div class="box">
     <h2>
       <a href="register.jsp">Register</a>
     </h2>
   </div>
```

```
  </div>

<!-- ____________________________________________________________________________ -->


<div id="footer">
  <h3>&copy; Copyrights Reserved<br>
                * Vinod * Vaibhav * Karthik * Rajesh * </h3>
</div>

</body>
</html>
```

Registration Module:

register.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merkle Hellman Knapsack Cryptosystem</title>
<link rel="stylesheet" type="text/css" href="mystyle.css"/>
</head>
<body>
        <div id="header">
                <h1>Image Encryption Using Merkle-Hellman Knapsack Cryptosystem</h1>
   </div>

<!-- ____________________________________________________________________________ -->

  <div id="main">
   <center>
   <form action="register" method="post">

   <table>
    <tr>
    <td>User Name </td>
```

```
    <td><input type="text" name="uname" required autofocus/></td>
    </tr>
    <tr>
    <td>Password </td>
    <td><input type="password" name="pass" required/></td>
    </tr>
    <tr>
    <td><input type="submit" value="Register"/> </td>
    <td><input type="reset" value="Reset"/></td>
    </tr>
   </table>
   </form>
   </center>

  </div>

<!-- ____________________________________________________________________________ -->


<div id="footer">
  <h3>&copy; Copyrights Reserved<br>
                  * Vinod * Vaibhav * Karthik * Rajesh * </h3>
</div>

</body>
</html>
```

Register.java

```
package com.mypack;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.merhell.ImgEnc;

/**
 * Servlet implementation class Register
 */
@WebServlet("/register")
public class Register extends HttpServlet {
	private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public Register() {
      super();
      // TODO Auto-generated constructor stub
   }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		// TODO Auto-generated method stub
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		// TODO Auto-generated method stub


		String uname = request.getParameter("uname");
		String pass = request.getParameter("pass");
		PrintWriter pw = response.getWriter();

		System.out.print("Uname :"+ uname +"\nPass :"+ pass);

		try {
```

```
                    Class.forName("oracle.jdbc.driver.OracleDriver");
                    Connection con = DriverManager.getConnection(
                                "jdbc:oracle:thin:@localhost:1521:xe", "img",
                                "img");
                    con.setAutoCommit(false);
                    PreparedStatement ps = con
                                .prepareStatement("insert into Users
values(U_id.nextval,?,?)");

                    ps.setString(1, uname);
                    ps.setString(2, pass);

                    ps.executeUpdate();
                    con.commit();
                    //-------------------------//
                    Statement st = con.createStatement(
                                ResultSet.TYPE_SCROLL_INSENSITIVE,
                                ResultSet.CONCUR_UPDATABLE);
                    ResultSet rs = st
                                .executeQuery("select max(U_id) as max from Users");
                    int i=0;
                    while (rs.next()) {
                     i= rs.getInt("max");

                    }
                    System.out.println("\n i  : "+ i);

                    ImgEnc ie = new ImgEnc();

                    int []prkey = ie.privatekey();

                    PreparedStatement ps2 = con.prepareStatement("insert into privatekey
values(?,?,?,?,?,?,?,?,?,?,?)");
                    ps2.setInt(1, i);
                    ps2.setInt(2, prkey[0]);
                    ps2.setInt(3, prkey[1]);
                    ps2.setInt(4, prkey[2]);
                    ps2.setInt(5, prkey[3]);
                    ps2.setInt(6, prkey[4]);
                    ps2.setInt(7, prkey[5]);
                    ps2.setInt(8, prkey[6]);
                    ps2.setInt(9, prkey[7]);
                    ps2.setInt(10, prkey[8]);
                    ps2.setInt(11, prkey[9]);
```

```
                        ps2.executeUpdate();
                        con.commit();
                        //----------------------------------------//
                        int []pukey = ie.publickey();

                        PreparedStatement ps1 = con.prepareStatement("insert into publickey
values(?,?,?,?,?,?,?,?,?)");
                        ps1.setInt(1, i);
                        ps1.setInt(2, pukey[0]);
                        ps1.setInt(3, pukey[1]);
                        ps1.setInt(4, pukey[2]);
                        ps1.setInt(5, pukey[3]);
                        ps1.setInt(6, pukey[4]);
                        ps1.setInt(7, pukey[5]);
                        ps1.setInt(8, pukey[6]);
                        ps1.setInt(9, pukey[7]);
                        ps1.executeUpdate();
                        con.commit();
                        //--------------------//

                        con.close();
                        ps.close();
                        ps1.close();
                        ps2.close();
                        st.close();
                        rs.close();
                        pw.println("<html><body>");
                        pw.print("<h2>Registration Successful</h2>\n");
                        pw.print("Click <a href=\"login.jsp\"> here </a> to login");
                        pw.println("</body></html>");
                } catch (Exception ex) {
                        pw.println(ex.getMessage());
                        System.out.println("\n"+ex.getMessage());
                } finally {
                        pw.close();

                }

        }

}
```

Login Module:

login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
	pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merkle Hellman Knapsack Cryptosystem</title>
<link rel="stylesheet" type="text/css" href="mystyle.css"/>
</head>
<body>
	<div id="header">
		<h1>Image Encryption Using Merkle-Hellman Knapsack Cryptosystem</h1>
  </div>

<!-- ____________________________________________________________________ -->

  <div id="main">
   <center>
   <form action="login" method="post">

   <table>
    <tr>
    <td>User Name </td>
    <td><input type="text" name="uname" required autofocus/></td>
    </tr>
    <tr>
    <td>Password </td>
    <td><input type="password" name="pass" required/></td>
    </tr>
    <tr>
    <td><input type="submit" value="Login"/> </td>
    <td><input type="reset" value="Reset"/></td>
    </tr>
   </table>
   </form>
   </center>

  </div>
```

```
<!-- _____________________________________________________________________ -->


<div id="footer">
  <h3>&copy; Copyrights Reserved<br>
                * Vinod * Vaibhav * Karthik * Rajesh * </h3>
</div>

</body>
</html>
```

Login.java

```
package com.mypack;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Login
 */
@WebServlet("/login")
public class Login extends HttpServlet {
        private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Login() {
```

```
    super();
    // TODO Auto-generated constructor stub
  }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                // TODO Auto-generated method stub
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                // TODO Auto-generated method stub

                String uname = request.getParameter("uname");
                String pass = request.getParameter("pass");
                PrintWriter pw = response.getWriter();

                try {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        Connection con = DriverManager.getConnection(
                                        "jdbc:oracle:thin:@localhost:1521:xe", "img",
                                        "img");
                        con.setAutoCommit(false);

                        Statement st = con.createStatement(
                                        ResultSet.TYPE_SCROLL_INSENSITIVE,
                                        ResultSet.CONCUR_UPDATABLE);

                        ResultSet rs = st
                                        .executeQuery("select U_id from users where u_name =
'"+ uname +"' and "
                                                        + "pass = '"+ pass+"'");

                        if (rs.next()){
                                Cookie cook1 = new Cookie("id",
String.valueOf(rs.getInt("U_id")));
                                Cookie cook2 = new Cookie("uname", uname);
```

```
                        cook1.setMaxAge(24*60*60);
                        cook2.setMaxAge(24*60*60);
                        response.addCookie(cook1);
                        response.addCookie(cook2);
                }

                con.commit();
                con.close();
                rs.close();
                st.close();
                response.sendRedirect("home.jsp");
                pw.close();
            } catch (Exception ex) {
                pw.println(ex.getMessage());
                System.out.println(ex.getMessage());
            } finally {
                pw.close();

            }
        }

}
```

home.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merkle Hellman Knapsack Cryptosystem</title>
<link rel="stylesheet" type="text/css" href="mystyle.css"/>
</head>
<body>
    <div id="header">
        <h1>Image Encryption Using Merkle-Hellman Knapsack Cryptosystem</h1>
  </div>

<!-- ____________________________________________________________________________ -->
<%
```

```
String uname="";
Cookie[] cookies = request.getCookies();
boolean foundCookie = false;

for(int i = 0; i < cookies.length; i++) {
   Cookie cookie1 = cookies[i];
   if (cookie1.getName().equals("uname")) {
      uname = cookie1.getValue();
      foundCookie = true;
   }
}

if (!foundCookie) {
   response.sendRedirect("login.jsp");
}

%>

  <div id="main">
  <center>
    <h2>Welcome <%=uname%></h2>
    <form action="logout" method="post">
    <input type="submit" value="  Logout  " >
    </form>
  </center>
  <br><br>
  <div class="box">
     <h2>
       <a href="send.jsp">Send Images</a>
     </h2>
   </div>
   <div class="box">
     <h2>
       <a href="view.jsp">View Images</a>
     </h2>
   </div>
  </div>

<!-- ____________________________________________________________________________ -->


<div id="footer">
  <h3>&copy; Copyrights Reserved<br>
```

```
                * Vinod * Vaibhav * Karthik * Rajesh * </h3>
</div>


</body>
</html>
```

Encryption Module:

send.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merkle Hellman Knapsack Cryptosystem</title>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
<body>
        <%
                String uname = "";
                Cookie[] cookies = request.getCookies();

                if (cookies.length == 0) {
                        response.sendRedirect("login.jsp");
                }
        %>
        <div id="header">
                <h1>Image Encryption Using Merkle-Hellman Knapsack Cryptosystem</h1>
        </div>

        <!--
_____________________________________________________________________________ -->

        <div id="main">
                <center>
                        <%
                                String name = "";
```

```
                        int id = 0;
                        for (int i = 0; i < cookies.length; i++) {
                                if (cookies[i].getName().equals("uname")) {
                                        name = cookies[i].getValue();
                                }
                                if (cookies[i].getName().equals("id")) {
                                        id = Integer.parseInt(cookies[i].getValue());
                                }
                        }
                %>
                <form method="post" action="sendimg" enctype="multipart/form-
data">
                <input type="hidden" name="send" value="<%=id%>"><br>
                <table>
                <tr>
                 <td>Select Recipient</td>
                 <td><select name="receive" required>
                                <option value="">** Select One **</option>
                                <%
                                        Class.forName("oracle.jdbc.driver.OracleDriver");
                                        Connection con = DriverManager.getConnection(

        "jdbc:oracle:thin:@localhost:1521:xe", "img", "img");
                                        try {
                                                Statement st = con.createStatement(

        ResultSet.TYPE_SCROLL_INSENSITIVE,

        ResultSet.CONCUR_UPDATABLE);
                                                ResultSet rs = st
                                                                .executeQuery("select
U_id,U_name from Users");
                                                while (rs.next()) {
                                %>
                                <option
value='<%=rs.getInt("U_id")%>'><%=rs.getString("U_name")%></option>
                                <%
                                        }
                                                st.close();
                                                rs.close();
                                                con.close();
                                        } catch (Exception e) {
                System.out.println(e.getMessage());
                                        }
```

```
                                        %>
                                    </select> </td>
                            </tr>
                        <tr>
                                    <td>Select Image to Encrypt and Send </td>
                                    <td> <input type="file" name="file"
                                        size="60" /></td>
                            </tr>
                            <tr>
                            <td><input type="submit" value="Send" /></td>
                            <td><input type="reset" value="Reset" /></td>
                            </tr>
                             </table>
                            </form>
                    </center>

        </div>

        <!--
_____________________________________________________________________________ -->


        <div id="footer">
 <h3>&copy; Copyrights Reserved<br>
                * Vinod * Vaibhav * Karthik * Rajesh * </h3>
</div>

</body>
</html>
```

SendImg.java

```
package com.mypack;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.List;

import javax.imageio.ImageIO;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;

import com.merhell.ImgEnc;

/**
 * Servlet implementation class SendImg
 */
@WebServlet("/sendimg")
public class SendImg extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private static int filename = 0;

	/**
	 * @see HttpServlet#HttpServlet()
	 */
	public SendImg() {
		super();
		// TODO Auto-generated constructor stub
	}

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
	 *      response)
	 */
	protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {
		// TODO Auto-generated method stub
	}

	/**
```

```
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
 *      response)
 */
protected void doPost(HttpServletRequest request,
                HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter pw = response.getWriter();

        try {
                DiskFileItemFactory factory = new DiskFileItemFactory();
                ServletFileUpload sfu = new ServletFileUpload(factory);

                if (!ServletFileUpload.isMultipartContent(request)) {
                        System.out.println("sorry. No image uploaded");
                        return;
                }
                // parse request
                List items = sfu.parseRequest(request);
                FileItem send = (FileItem) items.get(0);
                String sender = send.getString();

                FileItem receive = (FileItem) items.get(1);
                String receiver = receive.getString();

                System.out.println("sender : " + sender + " \nreceiver :"
                                + receiver);

                // get uploaded file
                FileItem fileitem = (FileItem) items.get(2);

                // String path = request.getServletContext().getRealPath("");
                InputStream is = fileitem.getInputStream();

                ImgEnc ie = new ImgEnc();
                BufferedImage src = ImageIO.read(is);
                int beta[] = new int[8];
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection con = DriverManager.getConnection(
                                "jdbc:oracle:thin:@localhost:1521:xe", "img", "img");
                Statement st = con.createStatement(
                                ResultSet.TYPE_SCROLL_INSENSITIVE,
                                ResultSet.CONCUR_UPDATABLE);
```

```
                        ResultSet rs = st
                                        .executeQuery("select B1,B2,B3,B4,B5,B6,B7,B8 from
publickey where U_id="
                                                        + receiver);

                        while (rs.next()) {
                                beta[0] = rs.getInt("B1");
                                beta[1] = rs.getInt("B2");
                                beta[2] = rs.getInt("B3");
                                beta[3] = rs.getInt("B4");
                                beta[4] = rs.getInt("B5");
                                beta[5] = rs.getInt("B6");
                                beta[6] = rs.getInt("B7");
                                beta[7] = rs.getInt("B8");
                        }

                        BufferedImage dst[] = ie.encryption(src, beta);

                        File outputimg = new File(
                                        "E:\\Java EE
Eclipse\\MyProjects\\FinalProject\\encrypted_images\\code_"
                                                        + filename + ".png");
                        ImageIO.write(dst[0], "png", outputimg);
                        File outputcode = new File(
                                        "E:\\Java EE
Eclipse\\MyProjects\\FinalProject\\encrypted_images\\image_"
                                                        + filename + ".png");
                        ImageIO.write(dst[1], "png", outputcode);

                        PreparedStatement ps = con.prepareStatement("insert into
message(sender_id,receiver_id,img_name) values(?,?,?)");
                        ps.setInt(1, Integer.parseInt(sender));
                        ps.setInt(2, Integer.parseInt(receiver));
                        ps.setString(3, String.valueOf(filename));
                        ps.executeUpdate();

                        filename++;

                        con.commit();
                        ps.close();
                        rs.close();
                        st.close();
                        con.close();
                        pw.println("<html><body>");
```

```
                        pw.print("<h2>Image Successfully Encrypted and sent</h2>\n");
                        pw.print("Click <a href=\"send.jsp\"> here </a> to Send another
Image\n\n");
                        pw.print("Click <a href=\"home.jsp\"> here </a> to navigate to home");
                        pw.println("</body></html>");

                } catch (Exception e) {
                        pw.print(e.getMessage() + "  Error Dude !!");
                        System.out.print("\n" + e.getMessage() + "  Error Dude !!");

                }
        }
}
```

Decryption Module:

view.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
        pageEncoding="ISO-8859-1"%>
<%@ page import="java.sql.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merkle Hellman Knapsack Cryptosystem</title>
<link rel="stylesheet" type="text/css" href="mystyle.css"/>
</head>
<body>
        <div id="header">
                <h1>Image Encryption Using Merkle-Hellman Knapsack Cryptosystem</h1>
   </div>

<!-- ____________________________________________________________________________ -->
<%

String uname="";
int id=0;
Cookie[] cookies = request.getCookies();
boolean foundCookie = false;
```

```
for(int i = 0; i < cookies.length; i++) {
   Cookie cookie1 = cookies[i];
   if (cookie1.getName().equals("uname")) {
      uname = cookie1.getValue();
      foundCookie = true;
   }
   if (cookie1.getName().equals("id")) {
      id = Integer.parseInt(cookie1.getValue());
      foundCookie = true;
   }
}

if (!foundCookie) {
   response.sendRedirect("login.jsp");
}

%>

  <div id="main">
  <center>
   <h2>Welcome <%=uname%></h2>
   <form action="logout" method="post">
   <input type="submit" value="  Logout  " >
   </form>
  </center>
  <br><br>
  <center>
   <table class="tab">
   <tr>
    <th>S.No</th>
    <th>Sender</th>
    <th>Date</th>
    <th> </th>
   </tr>
   <%try{
  Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection(
                        "jdbc:oracle:thin:@localhost:1521:xe", "img", "img");
        Statement st = con.createStatement(
                        ResultSet.TYPE_SCROLL_INSENSITIVE,
                        ResultSet.CONCUR_UPDATABLE);
        ResultSet rs = st.executeQuery("select U_name,Dt,Img_name from Users,Message
where Receiver_id = "+ id +
```

```
                        "and Sender_id=U_id order by Img_name DESC");
        int i=1;
        while(rs.next()){
  %>
   <tr>
   <td><%=i++ %></td>
    <td> <%=rs.getString("U_name") %></td>
    <td><%=rs.getDate("Dt") %> </td>
    <td>
      <form action="viewimg.jsp" method="get">
        <input type="hidden" name="fname" value='<%=rs.getString("Img_name")%>'>
        <input type="submit" value="View" >
      </form>
    </td>
   </tr>

   <%
        }

        st.close();
        rs.close();
        con.close();
   }catch(Exception e){
         System.out.print(e.getMessage());
   }
   %>
   </table>
   </center>

  </div>

<!-- ____________________________________________________________________________ -->

<br>
<div id="footer">
  <h3>&copy; Copyrights Reserved<br>
                * Vinod * Vaibhav * Karthik * Rajesh * </h3>
</div>


</body>
</html>
```

Decrypt.java

```
package com.mypack;

import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.OutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import javax.imageio.ImageIO;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.merhell.ImgEnc;

/**
 * Servlet implementation class Decrypt
 */
@WebServlet("/decrypt")
public class Decrypt extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private Connection con = null;
	private Statement st = null;
	private ResultSet rs = null;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Decrypt() {
        super();
        // TODO Auto-generated constructor stub
    }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
```

```
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		// TODO Auto-generated method stub
	}

	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		// TODO Auto-generated method stub
		String filename = request.getParameter("fname");
		ImgEnc ie = new ImgEnc();
		int id = 0;
		System.out.println("Running");
		Cookie[] cookies = request.getCookies();
		for (int i = 0; i < cookies.length; i++) {
			if (cookies[i].getName().equals("id")) {
				id = Integer.parseInt(cookies[i].getValue());
			}
		}
		int privatekey[]= new int[10];
		try{
			Class.forName("oracle.jdbc.driver.OracleDriver");
			con = DriverManager.getConnection(
					"jdbc:oracle:thin:@localhost:1521:xe", "img", "img");
				st = con.createStatement(
						ResultSet.TYPE_SCROLL_INSENSITIVE,
						ResultSet.CONCUR_UPDATABLE);
				rs = st.executeQuery("select S1,S2,S3,S4,S5,S6,S7,S8,Q,R from
privatekey where U_id= " +id);
				System.out.println("id: "+id);
				if(rs.next()) {
					privatekey[0]= rs.getInt("S1");
					privatekey[1]= rs.getInt("S2");
					privatekey[2]= rs.getInt("S3");
					privatekey[3]= rs.getInt("S4");
					privatekey[4]= rs.getInt("S5");
					privatekey[5]= rs.getInt("S6");
					privatekey[6]= rs.getInt("S7");
					privatekey[7]= rs.getInt("S8");
					privatekey[8]= rs.getInt("Q");
					privatekey[9]= rs.getInt("R");
```

```
                        }
                        System.out.println("q : "+privatekey[8]+"\nr : "+privatekey[9]);

                }catch(Exception e){
                        System.out.println(e.getMessage());
                }finally{
                        try{
                        rs.close();
                        st.close();
                        con.close();
                        }catch(Exception e){
                                System.out.print(e.getMessage());
                        }
                }
                response.setContentType("image/png");
                BufferedImage dec = ie.decryption(privatekey,filename);
                OutputStream out = response.getOutputStream();
                ImageIO.write(dec, "png", out);
                out.close();


        }

}
```

viewimg.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Merkle Hellman Knapsack Cryptosystem</title>
<link rel="stylesheet" type="text/css" href="mystyle.css"/>
</head>
<body>
        <div id="header">
                <h1>Image Encryption Using Merkle-Hellman Knapsack Cryptosystem</h1>
   </div>
```

```
<!-- ____________________________________________________________________________ -->
<%

String uname="";
Cookie[] cookies = request.getCookies();
boolean foundCookie = false;

for(int i = 0; i < cookies.length; i++) {
   Cookie cookie1 = cookies[i];
   if (cookie1.getName().equals("uname")) {
      uname = cookie1.getValue();
      foundCookie = true;
   }
}

if (!foundCookie) {
   response.sendRedirect("login.jsp");
}
  String fname = request.getParameter("fname");
%>

  <div id="main" style="height: auto;">
  <center>
    <h2>Welcome <%=uname%></h2>
    <form action="logout" method="post">
    <input type="submit" value="  Logout  " >
    </form>
  </center>
  <br><br>

   <center>
     <img style="border: 2px solid white;" height="50%" width="50%" alt="decrypted image"
src="decrypt?fname=<%=fname%>">
   </center><br><br>
   <center>
   <h3><a href="home.jsp">Home</a></h3>
   </center>
  </div>
                    <br>
<!-- ____________________________________________________________________________ -->


<div id="footer">
  <h3>&copy; Copyrights Reserved<br>
```

```
                * Vinod * Vaibhav * Karthik * Rajesh * </h3>
</div>

</body>
</html>
```

Logout Module:

Logout.java

```
package com.mypack;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Logout
 */
@WebServlet("/logout")
public class Logout extends HttpServlet {
        private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Logout() {
        super();
        // TODO Auto-generated constructor stub
    }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
```

```
                // TODO Auto-generated method stub
        }

        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                // TODO Auto-generated method stub

                response.setContentType("text/html");
      Cookie idCookie = null;
      Cookie unameCookie = null;
      Cookie[] cookies = request.getCookies();
      if(cookies != null){
      for(Cookie cookie : cookies){
         if(cookie.getName().equals("id")){
            idCookie = cookie;
         }
         if(cookie.getName().equals("uname")){
            unameCookie = cookie;
         }
      }

      if(idCookie != null){
         idCookie.setMaxAge(0);
         response.addCookie(idCookie);
      }
      if(unameCookie != null){
         unameCookie.setMaxAge(0);
         response.addCookie(unameCookie);
      }
      response.sendRedirect("index.jsp");
   }
         }

}
```

Database:

Tables Description:

```
SQL>
SQL> desc users
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 U_ID                                      NOT NULL NUMBER(2)
 U_NAME                                             VARCHAR2(15)
 PASS                                               VARCHAR2(20)

SQL> desc privatekey
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 U_ID                                               NUMBER(2)
 S1                                        NOT NULL NUMBER(3)
 S2                                        NOT NULL NUMBER(3)
 S3                                        NOT NULL NUMBER(3)
 S4                                        NOT NULL NUMBER(3)
 S5                                        NOT NULL NUMBER(3)
 S6                                        NOT NULL NUMBER(3)
 S7                                        NOT NULL NUMBER(3)
 S8                                        NOT NULL NUMBER(3)
 Q                                         NOT NULL NUMBER(3)
 R                                         NOT NULL NUMBER(3)

SQL> desc publickey
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 U_ID                                               NUMBER(2)
 B1                                        NOT NULL NUMBER(3)
 B2                                        NOT NULL NUMBER(3)
 B3                                        NOT NULL NUMBER(3)
 B4                                        NOT NULL NUMBER(3)
 B5                                        NOT NULL NUMBER(3)
 B6                                        NOT NULL NUMBER(3)
 B7                                        NOT NULL NUMBER(3)
 B8                                        NOT NULL NUMBER(3)

SQL> desc message
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------

 SENDER_ID                                 NOT NULL NUMBER(2)
 RECEIVER_ID                               NOT NULL NUMBER(2)
 IMG_NAME                                           VARCHAR2(15)
 DT                                        NOT NULL DATE

SQL>
SQL>
```

7. TESTING

The number of changing pixel rate (NPCR) is the most common quantity used to evaluate the strength of image encryption algorithms/ciphers with respect to differential attacks.

NPCR:

NPCR become the widely used security analyses in the image encryption community for differential attacks. Suppose ciphertext images before and after one pixel change in a plaintext image are C^1 and C^2 , respectively; the pixel value at grid (i,j) in C^1 and C^2 are denoted as C^1 (i,j) and C^2 (i,j); and a bipolar array is defined in Eqn. (1). Then the NPCR can be mathematically defined by Eqn. (2) where symbol *T* denotes the total number pixels in the ciphertext, symbol *F* denotes the largest supported pixel value compatible with the ciphertext image format.

$$D(i,j) = \begin{cases} 0, if\, C^1(i,j) = C^2(i,j) \\ 1, if\, C^1(i,j) \neq C^2(i,j) \end{cases} \qquad (1)$$

$$\text{NPCR: } \mathcal{N}(C^1, C^2) = \sum_{i,j} \frac{D(i,j)}{T} \times 100\% \qquad (2)$$

It is clear that NPCR concentrates on the absolute number of pixels which changes value in differential attacks. The range of NPCR is [0, 1]. When $N(C^1, C^2) = 0$, it implies that all pixels in C^2 remain the same values as in C^1. When $N(C^1, C^2) = 1$, it implies that all pixel values in C^2 are changed compared to those in C^1. In other words, it is very difficult to establish relationships between this pair of ciphertext image C^1 and C^2. However, $N(C^1, C^2) = 1$ rarely happens, because even two independently generated true random images fail to achieve this NPCR maximum with a high possibility, especially when the image size is fairly large compared to *F*.

8. SCREENSHOTS

index.jsp

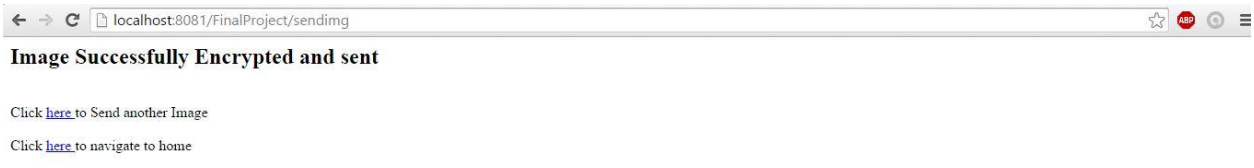
register.jsp

login.jsp

home.jsp

send.jsp

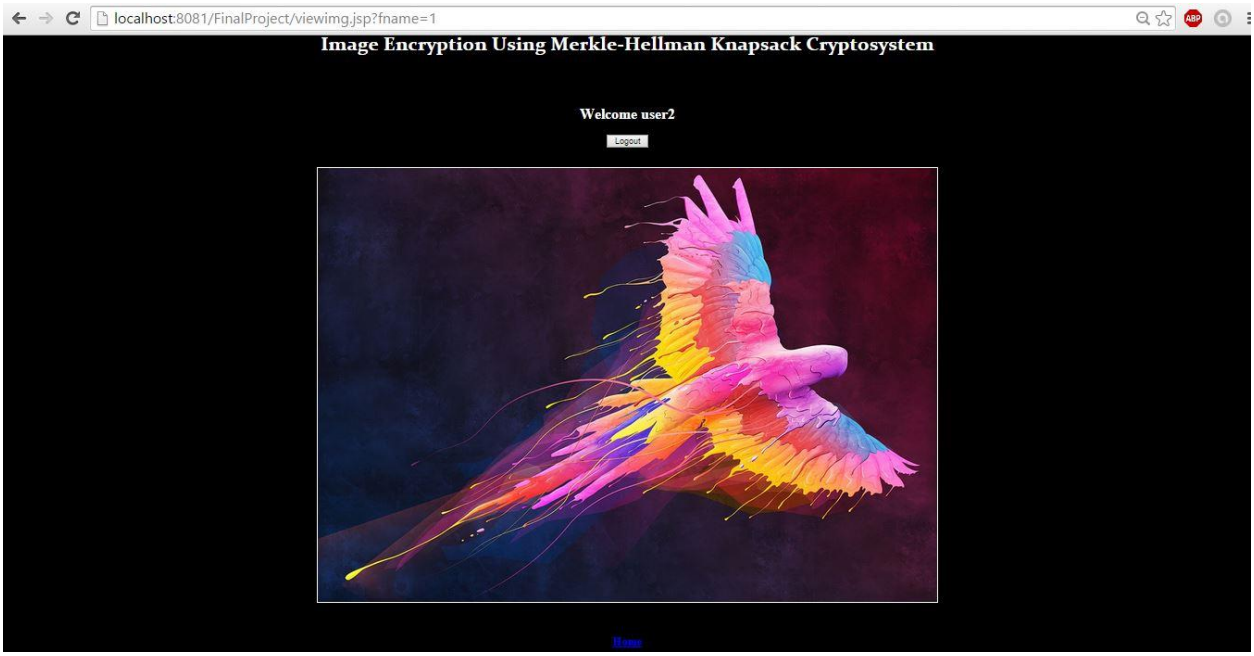
sendimg



Server Directory:

view.jsp

viewimg.jsp


localhost:8081/FinalProject/viewimg.jsp?fname=1
Image Encryption Using Merkle-Hellman Knapsack Cryptosystem
Welcome user2
Logout
Home

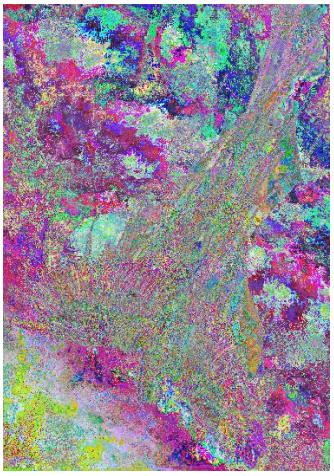
Sample Database Values:

```
SQL>
SQL>
SQL> select * from users;

      U_ID U_NAME          PASS
---------- --------------- --------------------
        37 user1           user1
        38 user2           user2

SQL> select * from privatekey;

      U_ID         S1         S2         S3         S4         S5         S6
---------- ---------- ---------- ---------- ---------- ---------- ----------
        S7         S8          Q          R
---------- ---------- ---------- ----------
        37          1          3          6         12         24         48
        96        192        996        361

        38          2          3          6         13         25         51
       101        202        488        401


SQL> select * from publickey;

      U_ID         B1         B2         B3         B4         B5         B6
---------- ---------- ---------- ---------- ---------- ---------- ----------
        B7         B8
---------- ----------
        37        361         87        174        348        696        396
       792        588

        38        314        227        454        333        265        443
       485        482


SQL> select * from message;

 SENDER_ID RECEIVER_ID IMG_NAME        DT
---------- ----------- --------------- ---------
        37          38 1               29-MAR-15

SQL>
SQL>
```

9. RESULTS

Original Image

Encrypted Image



Decrypted Image

10. Conclusion

We introduced a new approach for image encryption using Merkel Hellman cryptosystem which is an emerging technology used for owner ship assertion. Experimental results evaluated for image size of 32 X 32. The evaluation scheme can be extended for images of larger sizes. In the evaluation process, each pixel is encrypted by considering the same super increasing sequence. To meet requirement of confidentiality to the maximum extent each pixel of the image is to be encrypted with different super increasing sequence.

11. Future Scope

The future of the proposed scheme is that it can extended for encrypting the video messages as well as sound encryption process.

12. REFERENCES

Java – www.sun.com

Wikipedia - www.wikipedia.com

Oracle – www.oracle.com

Visual Paradigm – www.visual-paradigm.com

OTHER REFERENCES:

www.google.com

www.tutorialspoint.com

www.stackoverflow.com

www.cyberjournals.com