**A project report on**

Image Authentication over wireless channels

Project report submitted in partial fulfillment of the requirement for the award of the Degree of B.Tech

By

Sahitya.N Vijetha.B Chinababu.T

(06A81A1241) (06A81A1258) (06A81A1249)

Under guidance of

Mrs A.L.Lavanya

Assistant Professor

****

**Department of Information Technology**

**Sri Vasavi Engineering College**

**(Affiliated to Jawaharlal Nehru Technological University)**

**Kakinada.**

**Tadepalligudem, W.G.DT**

**Andhra Pradesh, 2010**

**A project report on**

Image Authentication over wireless channels

Project report submitted in partial fulfillment of the requirement for the award of the Degree of B.Tech

By

Sahitya.N Vijetha.B Chinababu.T

(06A81A1241) (06A81A1258) (06A81A1249)

Under guidance of

Mrs A.L.Lavanya

Assistant Professor



**Department of Information Technology**

**Sri Vasavi Engineering College**

**(Affiliated to Jawaharlal Nehru Technological University)**

**Kakinada.2010**

**Sri Vasavi Engineering College**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**CERTIFICATE**

This is to certify that the project report entitled **“IMAGE AUTHENTICATION OVER WIRELESS CHANNELS”** being submitted by **N.Sahitya, B.Vijetha, T.ChinaBabu** in partial fulfillment for the award of **BACHELOR OF TECHNOLOGY** in **SRI VASAVI ENGINEERING COLLEGE**  to the Jawaharlal Nehru Technological University is a record of bonafied work carried out by him under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or institute for the award of any Degree or diploma.

**HEAD OF THE DEPARTMENT PROJECT GUIDE**  **(Sir G.Srinivasa Rao) (A.L.Lavanya)**

**EXTERNAL EXAMINAR**

**ACKNOWLEDGEMENT**

We are graceful to our H.O.D Sri G.SrinivasaRao garu, M.Tech for his esteemed guidance. We are thankful to our internal guide A.L.Lavanya, M.Tech who helped us in doing project successfully. We also take this opportunity to thank each and every person who helped in completing the project.

An Endeavour over long period can also be successful by constant effort and encouragement. We wish to take this opportunity to express our deep gratitude to all the people who have extended their cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

**N.Sahitya**

**B.Vijetha**

**T.ChinaBabu**

**ABSTRACT**

The introduction of 3G wireless communication systems, together with the invasive distribution of digital images and the growing concern on their originality triggers an emergent need of authenticating images received by unreliable channels, such as public Internet and wireless networks. To meet this need, a content-based image authentication scheme that is suitable for an insecure network and robust to transmission errors is proposed. The proposed scheme exploits the scalability of a structural digital signature in order to achieve a good tradeoff between security and image transfer for networked image applications. In this scheme, multi-scale features are used to make digital signatures robust to image degradations and key dependent parametric wavelet filters are employed to improve the security against forgery attacks. This scheme is also able to distinguish tampering areas in the attacked image. Experimental results show the robustness and validity of the proposed scheme.

Table of Contents

Table of Contents

1. Introduction 1

1.1 General 1

1.2 ExistedSystem 2

1.3 Proposed System 2

1.4 Purpose 2

1.5 Project Scope 2

2. Software and Hardware requirements 3

2.1 Software Requirement specification 3

2.2 Hardware Requirement specification 3

3. Literature Survey 4

3.1 Technical Feasibility 4

3.2 Operational Feasibility 4

3.3 Economic Feasibility 5

4.Software Requirement Analysis 6

4.1 System Analysis 6

4.2 Module Description 7

4.2 System Architecture 9

5. Software Design 10

5.1 DataFlowDiagram 10

5.2 UML Diagrams 12

6. Coding 25

7. Testing 51

7.1 Black Box Testing 51

7.2 White Box Testing 52

7.3 Unit Testing 53

7.4 Integration Testing 53

7.5 Functional Testing 53

7.6 System Testing 54

8. Output Screens 55

9. Conclusion 70

10. Further Enhancement 71

11. References 72

12. Appendices. 73

**List of Abbreviations**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **DESCRIPTION** |
| 1 | Quos | Quality of Service |
| 2 | DFD | Data Flow Diagram |
| 3 | API | Application Programming Interface |
| 4 | UML | Unified Modeling Language |
| 5 | GUI | Graphical User Interface |
| 6 | IDE | Integrated Development Environment |
| 7 | GS | Guaranteed service |
| 8 | EF | Expedited Forwarding |
| 9 | AF | Assured Forwarding |
| 10 | BE | Best Effort |
| 11 | SDLC | Software Development Life Cycle |
| 12 | JPEG | Joint Photographic Experts Group |

**List of Symbols**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **SYMBOL NAME** | **SYMBOL** | **DESCRIPTION** |
| 1 | Class |  | Classes represent a collection of similar entities grouped together. |
| 2 | Association |  | Association represents a static relationship between classes. |
| 3 | Aggregation |  | Aggregation is a form of association. It aggregates several classes into single class. |
| 4 | Actor |  | Actors are the users of the system and other external entity that react with the system. |
| 5 | Use Case |  | A use case is a interaction between the system and the external environment. |
| 6 | Relation (Uses) |  | It is used for additional process communication. |
| 7 | Communication |  | It is the communication between various use cases. |
| 8 | State |  | It represents the state of a process. Each state goes through various flows. |
| 9 | Initial State |  | It represents the initial state of the object. |
| 10 | Final State |  | It represents the final state of the object. |
| 11 | Control Flow |  | It represents the various control flow between the states. |
| 12 | Decision Box |  | It represents the decision making process from a constraint. |
| 13 | Component |  | Components represent the physical components used in the system. |
| 14 | Node |  | Deployment diagrams use the nodes for representing physical modules, which is a collection of components. |
| 15 | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 16 | External Entity |  | It represent any external entity such as keyboard, sensors etc which are used in the system. |
| 17 | Transition |  | It represent any communication that occurs between the processes. |
| 18 | Object Lifeline |  | Object lifelines represents the vertical dimension that objects communicates. |
| 19 | Message |  | It represents the messages exchanged. |

**List of Figures**

**Fig No Title Page No**

4.2 Architecture 9

5.1 Data flow diagram 11

5.2.1 Use case Diagram 13

5.2.2 Class Diagram 16

5.2.3 Sequence Diagram 17

5.2.4 Collaboration Diagram 19

5.2.5 Activity Diagram 21

5.2.6 Component diagram 23

5.2.7 Deployment Diagram 24

**1. INTRODUCTION**

**1.1 General**

Recent advances in networking and digital media technologies have created a large number of networked multimedia applications. Those applications are often deployed in a distributed network environment that makes multimedia contents vulnerable to privacy and malicious attacks. For insecure environments, it is possible for an enemy to tamper with images during transmission. To guarantee trustworthiness, image authentication techniques have emerged to confirm content integrity and prevent forgery. These techniques are required to be robust against normal image processing and transmission errors, while being able to detect malevolent tampering on the image. Such authentication techniques have wide applicability in law, commerce, journalism and national defence. In the literatures, methods of image content authentication can be categorized into either digital signature based or watermarking based.

To provide a strong authentication for transmitting multimedia image data securely we have new image authentication technique which is digital signature based or water marking based .A digital signature also called crypto-hash value captures image content in compact representation and stores in a file. From this file it uses image for authentication. Now, signature based method provide integrity of image and also prevents repudiation of the sender. Normally, during data transmission of images there is lossy of image information particularly in wireless network. To prevent this content based digital signature approach is useful. It verifies image content bit by bit effectively accurately.

The major differences that differentiate the proposed scheme from existing state-of-the-art approaches are: (1) it works at a semi-fragile level, which means that some manipulations on the image will be considered acceptable; (2) more robustness – it can tolerate a range of attacks while accurately locating the tampered area – is achieved by exploiting the concept of structural digital signature (SDS); (3) the integration of the SDS and key dependent parametric wavelet filters makes the scheme more efficient to security attacks; (4) the computation complexity is reduced because of the framework of a lifting-based wavelet transform; and (5) the ability to support efficient and accurate tamper localization in spite of information loss in large areas or high variant areas.

**1.2 Existing System**

Networked multimedia applications are often deployed in a distributed network environment that makes multimedia contents vulnerable to privacy and malicious attacks. For insecure environments, it is possible for an enemy to tamper with images during transmission. To guarantee trustworthiness, image authentication techniques have emerged to confirm content integrity and prevent forgery. These techniques are required to be robust against normal image processing and transmission errors, while being able to detect malevolent tampering on the image.

**1.3 Proposed System**

The proposed scheme exploits the scalability of a structural digital signature in order to achieve a good tradeoff between security and image transfer for networked image applications. In this scheme, multi-scale features are used to make digital signatures robust to image degradations and key dependent parametric wavelet filters are employed to improve the security against forgery attacks. This scheme is also able to distinguish tampering areas in the attacked image.

**1.4 Purpose**

The main purpose of the robust and secure digital signature scheme for image authentication is to provide security for multimedia images during transmission in wireless network. It’s objective is to prevent tampering of multimedia images.

## Project Scope

The scope of this system is to provide efficient accurate transmission of multimedia data from client to server in a heterogeneous distributed wireless network. When client and server executing multimedia applications. By applying digital signature methods it provides image authentication, it provides integrity to image at both client and server area, prevents malicious and mage during transmission, prevents repudiation of sender from client.

# 2. SOFTWARE AND HARDWARE REQUIREMENTS

# 2.1 Software requirement Specification

# 

* Microsoft Windows XP Professional.
* JDK 6.0.

## 2.2 Hardware Requirement Specification

* Pentium 4 processor.
* 1 GB RAM.
* 80 GB Hard Disk Space

**3. Literature Survey**

**Introduction**

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it’s worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

**3.1 Technical Feasibility**

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis. Understand the different technologies involved in the proposed system before commencing the project we have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Is the required technology available with the organization?

**3.2 Operational Feasibility**

Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

* Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
* Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
* Have the user been involved in the planning and development of the project?
* Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible.

**3.3 Economic Feasibility**

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality better decision making timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale.

**4. SOFTWARE REQUIREMENT ANALYSIS**

**4.1 System Analysis**

The first step in developing anything is to state the requirements. This applies just as much to leading edge research as to simple programs and to personal programs, as well as to large team efforts. Being vague about your objective only postpones decisions to a later stage where changes are much more costly.

The problem statement should state what is to be done and not how it is to be done. It should be a statement of needs, not a proposal for a solution. A user manual for the desired system is a good problem statement. The requestor should indicate which features are mandatory and which are optional, to avoid overly constraining design decisions. The requestor should avoid describing system internals, as this restricts implementation flexibility. Performance specifications and protocols for interaction with external systems are legitimate requirements. Software engineering standards, such as modular construction, design for testability, and provision for future extensions, are also proper.

Many problems statements, from individuals, companies, and government agencies, mixture requirements with design decisions. There may sometimes be a compelling reason to require a particular computer or language; there is rarely justification to specify the use of a particular algorithm. The analyst must separate the true requirements from design and implementation decisions disguised as requirements. The analyst should challenge such pseudo requirements, as they restrict flexibility. There may be politics or organizational reasons for the pseudo requirements, but at least the analyst should recognize that these externally imposed design decisions are not essential features of the problem domain.

A problem statement may have more or less detail. A requirement for a conventional product, such as a payroll program or a billing system, may have considerable detail. A requirement for a research effort in a new area may lack many details, but presumably the research has some objective, which should be clearly stated.

Most problem statements are ambiguous, incomplete, or even inconsistent. Some requirements are just plain wrong. Some requirements, although precisely stated, have unpleasant consequences on the system behavior or impose unreasonable implementation costs. Some requirements seem reasonable at first but do not work out as well as the request or thought. The problem statement is just a starting point for understanding the problem, not an immutable document. The purpose of the subsequent analysis is to fully understand the problem and its implications. There is no reasons to expect that a problem statement prepared without a fully analysis will be correct.

The analyst must work with the requestor to refine the requirements so they represent the requestor’s true intent. This involves challenging the requirements and probing for missing information. The psychological, organizational, and political considerations of doing this are beyond the scope of this book, except for the following piece of advice: If you do exactly what the customer asked for, but the result does not meet the customer’s real needs, you will probably be blamed anyway.

**4.2 Module Description**

This system is divided into three modules:

1. Network module.
2. Encryption module.
3. Decryption module.
4. Authentication module

**Network module**

This module is both accessed by sender and receiver in order to establish a wireless channel between them. During this process sender identifies the address of receiver. This module support following services:

1. Finding IP address.

2. Establish connection.

**Encryption module**

This module is accessed by sender at the location of sender side and its objective is to encrypt the plain text data given by sender. The encryption can be done either private or public keys or using water marking methods or combination it support following services:

1. Get input data.
2. Generate private or public key.
3. Select image
4. Encrypt using keys.
5. Encrypt by hiding data behind the image.

**Decryption module**

This module accessed by the receiver at receiver location and its objective is to copy decrypt the encrypted data to original form. During process it follows services:

1. Receive encrypted data.
2. Generate private or public key.
3. Decrypt by using keys.
4. Decrypt by extracting data from image

**Authentication module**

This is accessed by both sender and receiver. The objective of this module is the sender authenticate, encrypt data, receiver also authenticate encrypted data. So the sender ensures that data is received only by receiver.

**4.3 System Architecture**

Here, Client (sender) contain modules of Network module, encrypt module, authentication module. Server (receiver) contains decryption module, authenticate module.

The architecture is represented as follows:

**Client Server**

Decryption

Authenticate

Network module

Encrypt Module

Authenticate module

**Network**

**Figure 4.3 Architecture**

**5. SOFTWARE DESIGN**

**5.1 Data flow diagram**

A **data-flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system. DFD’s can also be used for the visualization of data processing.

The data flow diagram represents the node moving from sender to receiver. Sender gets a plain text and generates public or private keys and selects the image and hide the data. Encrypt the data and identifies receiver address and establish a connection. After establishing connection send the encrypted data. Receiver receives encrypted data and decrypts the data by generating keys. Decrypt the data finally displays plain text to the user.

Hence, flow of data from server to receiver is described for image authentication over wireless channels for heterogeneous system.

The diagrammatic representation is as follows:

**Figure 5.1 Data flow diagram**

**5.2 UML Diagrams**

**5.2.1 Use Case Diagram**

Since mainly provide security which is robust for images in wireless channels. Image is authenticating while it is sending. Security is provided by digital signature scheme. Here encryption is done either using private or public key water marking method.

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system’s behavior.

Actors in use case diagram are:

1. Sender
2. Receiver

Sender run number of nodes in a network. Sender selects the image which has to send to the receiver. Generate keys (private, public keys).Finally identifies destination and encrypt the data and then sends to receiver by encrypting the image using digital signature.

Receiver receives the encoded data by wavelet decomposition Decrypt the signature and perform content based verification.

**IDENTIFYING USE CASES:**

1. Run N number of nodes
2. Generate public and private key.
3. Neighbor discovery
4. Enter the destination.
5. Path discovery
6. Shortest path finding
7. Select image.
8. Encode the data.
9. Send
10. Receive encoded data
11. Wavelets decompose.
12. Decrypt signature.
13. Extract signature.
14. Content based verification.

The use case diagram is as follows:



**Figure 5.2.1a Use case diagram for sender**



**Figure 5.2.1b Use case diagram for receiver**

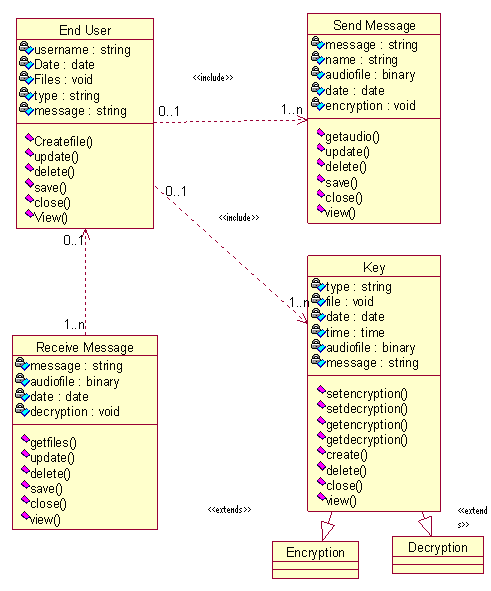
**5.2.2 CLASS DIAGRAM**

UML Class diagram shows the static structure of the model. The class diagram is a collection of static modeling elements, such as classes and their relationships, connected as a graph to each other and to their contents

The classes in this are:

1. Authentication
2. Data
3. Receiver.
4. Sender
5. Channels.
6. Action.
7. Key Pairs.

Diagram is as follows:

****

**Figure 5.2.2 Class diagram**

**5.2.3 SEQUENCE DIAGRAM**

Sequence diagram are an easy and intuitive way of describing the behavior Of a system by viewing the interaction between the system and its environment. A Sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time; the horizontal Dimension represents different objects. The vertical line is called is the object’s life line. The lifeline represents the object’s existence during the interaction.



**Figure 5.2.3a Sequence Diagram for sender**



**Figure 5.2.3b Sequence diagram for receiver**

**5.2.4 COLLABORATION DIAGRAM**

The collaboration diagram represents a collaboration, which is a set of objects Related in a particular context, and interaction, which is a set of messages exchanged among the objects within the collaboration to achieve a designed Outcome.

****

**Figure 5.2.4a Collaboration diagram for sender**

****

**Figure 5.2.4b Collaboration diagram for receiver**

**5.2.5 Activity Diagram**

The purpose of activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. Activity diagram can also be used to represent a class’s method implementation. A token represents an operation. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of activity symbol indicates a transition triggered by the completion.



**Figure 5.2.5a Activity diagram for Sender**



**Figure 5.2.5b Activity diagram for Receiver**

**5.2.6 COMPONENT DIAGRAM**

A component diagram in the Unified Modeling Language, depicts how components are wired together to form larger components and or software systems. Components are wired together by using an assemblyconnector to connect the required interface of one component with the provided interface of another component. Components diagrams can be used to illustrate the structure of arbitrarily complex systems.

****

**Figure 5.2.6 Component Diagram**

**5.2.7 DEPLOYMENT DIAGRAM**

A deployment **diagram** in the Unified Modeling Language serves to model the deployment of artifacts on deployment targets. Deployment diagrams show "the allocation of Artifacts to Nodes according to the Deployments defined between them.” Deployment of an artifact to a node is indicated by placing the artifact inside the node. Instances of nodes (and devices and execution environments) are used in deployment diagrams to indicate multiplicity of these nodes. For example, instances of an application execution environment may be deployed inside a single device node to represent application server clustering.



**Figure 5.2.7 Deployment diagram**

1. **CODING**

**Channel class**

package com.design;

import java.awt.FileDialog;

import java.awt.Image;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Properties;

import java.util.Vector;

import javax.imageio.ImageIO;

import javax.swing.ImageIcon;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import javax.swing.table.DefaultTableModel;

import org.jvnet.substance.SubstanceLookAndFeel;

import com.dwt.Action;

import com.dwt.Data;

import com.dwt.Receive;

import com.dwt.VerificationCode;

import com.ecc.Encryption;

/\*\*

\*

\* @author \_\_USER\_\_

\*/

public class Chennal extends javax.swing.JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

public Action action;

public String source;

/\*\* Creates new form Chennal \*/

public Chennal() {

initComponents();

init();

}

private void init() {

// TODO Auto-generated method stub

action = new Action();

source = action.getSource();

setTitle(source);

int dis = action.getDistance();

action.setProperty("Distance.properties", source, "" + dis);

txtPublicKey.setText(action.getPublicKey());

txtPrivateKey.setText(action.getPrivateKey());

lblSource.setText(source);

int port = action.getPort();

action.setProperty("Publickey.properties", source, ""

+ txtPublicKey.getText());

action.setProperty("Ports.properties", source, "" + port);

new Receive(this, port, action);

setFlag();

}

private void setFlag() {

// TODO Auto-generated method stub

try {

Properties port = new Properties();

port.setProperty("attack", "false");

FileOutputStream fos = new FileOutputStream("Attacker.properties");

port.store(fos, "Attack");

fos.close();

} catch (Exception e) {

// TODO: handle exception

e.printStackTrace();

}

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

// GEN-BEGIN:initComponents

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

SName = new javax.swing.JLabel();

lblSource = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jScrollPane1 = new javax.swing.JScrollPane();

jtaNeigh = new javax.swing.JTextArea();

btnNeigh = new javax.swing.JButton();

jLabel5 = new javax.swing.JLabel();

txtDestination = new javax.swing.JTextField();

btnPathDiscovery = new javax.swing.JButton();

jScrollPane2 = new javax.swing.JScrollPane();

tblPaths = new javax.swing.JTable();

jLabel6 = new javax.swing.JLabel();

lblShortestPath = new javax.swing.JLabel();

btnSelectImage = new javax.swing.JButton();

lblImage = new javax.swing.JLabel();

jLabel9 = new javax.swing.JLabel();

txtPublicKey = new javax.swing.JTextField();

jLabel10 = new javax.swing.JLabel();

txtPrivateKey = new javax.swing.JTextField();

btnEncryption = new javax.swing.JButton();

jLabel11 = new javax.swing.JLabel();

txtSignature = new javax.swing.JTextField();

btnSend = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

getContentPane().setLayout(null);

jLabel1.setFont(new java.awt.Font("Agency FB", 1, 36));

jLabel1.setForeground(new java.awt.Color(102, 0, 0));

jLabel1.setText("Digital Signature Scheme for Image Authentication");

getContentPane().add(jLabel1);

jLabel1.setBounds(40, 10, 610, 40);

SName.setText("Source:");

getContentPane().add(SName);

SName.setBounds(30, 70, 70, 14);

lblSource.setFont(new java.awt.Font("Tahoma", 1, 11));

lblSource.setText("lblSrc");

getContentPane().add(lblSource);

lblSource.setBounds(80, 70, 130, 14);

jLabel4.setText("Neighbours:");

getContentPane().add(jLabel4);

jLabel4.setBounds(30, 90, 110, 14);

jtaNeigh.setColumns(20);

jtaNeigh.setRows(5);

jScrollPane1.setViewportView(jtaNeigh);

getContentPane().add(jScrollPane1);

jScrollPane1.setBounds(30, 110, 240, 96);

btnNeigh.setText("Find Neighbour");

btnNeigh.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnNeighActionPerformed(evt);

}

});

getContentPane().add(btnNeigh);

btnNeigh.setBounds(30, 210, 240, 23);

jLabel5.setText("Enter the Destination Name:");

getContentPane().add(jLabel5);

jLabel5.setBounds(30, 240, 230, 14);

getContentPane().add(txtDestination);

txtDestination.setBounds(30, 260, 240, 20);

btnPathDiscovery.setText("Path Discovery");

btnPathDiscovery.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnPathDiscoveryActionPerformed(evt);

}

});

getContentPane().add(btnPathDiscovery);

btnPathDiscovery.setBounds(30, 290, 240, 23);

dftQue = new DefaultTableModel();

tblPaths.setModel(dftQue);

jScrollPane2.setViewportView(tblPaths);

dftQue.addColumn("Routes");

getContentPane().add(jScrollPane2);

jScrollPane2.setBounds(30, 320, 240, 110);

jLabel6.setText("Shortest Path:");

getContentPane().add(jLabel6);

jLabel6.setBounds(30, 440, 240, 14);

lblShortestPath.setFont(new java.awt.Font("Tahoma", 1, 11));

lblShortestPath.setText("SPath");

getContentPane().add(lblShortestPath);

lblShortestPath.setBounds(30, 460, 360, 14);

btnSelectImage.setText("Select Image");

btnSelectImage.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnSelectImageActionPerformed(evt);

}

});

getContentPane().add(btnSelectImage);

btnSelectImage.setBounds(300, 70, 270, 23);

getContentPane().add(lblImage);

lblImage.setBounds(300, 100, 270, 130);

jLabel9.setText("Public Key:");

getContentPane().add(jLabel9);

jLabel9.setBounds(300, 240, 70, 14);

getContentPane().add(txtPublicKey);

txtPublicKey.setBounds(300, 260, 330, 20);

jLabel10.setText("Private Key:");

getContentPane().add(jLabel10);

jLabel10.setBounds(300, 290, 70, 14);

getContentPane().add(txtPrivateKey);

txtPrivateKey.setBounds(300, 310, 330, 20);

btnEncryption.setText("Encryption");

btnEncryption.addActionListener(new java.awt.event.ActionListener() { public void actionPerformed(java.awt.event.ActionEvent evt) {

btnEncryptionActionPerformed(evt);

}

});

getContentPane().add(btnEncryption);

btnEncryption.setBounds(300, 340, 120, 23);

jLabel11.setText("Crypto Signature:");

getContentPane().add(jLabel11);

jLabel11.setBounds(300, 370, 120, 14);

getContentPane().add(txtSignature);

txtSignature.setBounds(300, 390, 330, 20);

btnSend.setText("Send");

btnSend.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

btnSendActionPerformed(evt);

}

});

getContentPane().add(btnSend);

btnSend.setBounds(300, 420, 110, 30);

setSize(650, 520);

setResizable(false);

setVisible(true);

}// </editor-fold>

// GEN-END:initComponents

String path;

private void openImage() {

// TODO Auto-generated method stub

FileDialog fd = new FileDialog(this, "Encoding", FileDialog.LOAD);

fd.setVisible(true);

String fff = fd.getDirectory();

if (fff != null) {

try {

File f = new File(fd.getDirectory(), fd.getFile());

System.out.println(f.getAbsolutePath());

BufferedImage image = ImageIO

.read(new File(f.getAbsolutePath()));

Image icon = image.getScaledInstance(270, 130,

Image.SCALE\_SMOOTH);

ImageIcon im = new ImageIcon(icon);

lblImage.setText("");

lblImage.setIcon(im);

path = f.getAbsolutePath();

} catch (IOException e) {

e.printStackTrace();

}

}

}

private void btnNeighActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

jtaNeigh.setText("");

action.setNeighbour(jtaNeigh, source);

}

private void btnPathDiscoveryActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

if (jtaNeigh.getText().equals(""))

JOptionPane.showMessageDialog(null,

"Neighbour Nodes are not there.");

else if (txtDestination.getText().equals(""))

JOptionPane.showMessageDialog(null, "Enter the Destination");

else {

Vector<String> path = new Vector<String>();

path.add(source);

action.routing(path, txtDestination.getText());

}

}

private void btnSendActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

Data data = new Data();

data.data = imgb;

data.file = new File(path).getName();

data.sign = new Encryption().ecies\_ex(txtSignature.getText());

action.sendData(txtDestination.getText(), data);

}

private void btnEncryptionActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

try {

File file = new File(path);

FileInputStream stream = new FileInputStream(file);

imgb = new byte[stream.available()];

stream.read(imgb);

String str = new String(imgb);

String code = VerificationCode.calculateCode(str, str);

txtSignature.setText(code);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

private void btnSelectImageActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

openImage();

}

/\*\*

\* @param args

\* the command line arguments

\*/

public static void main(String args[]) {

JFrame.setDefaultLookAndFeelDecorated(true);

SubstanceLookAndFeel

.setCurrentTheme("org.jvnet.substance.theme.SubstanceRaspberryTheme");

SubstanceLookAndFeel

.setCurrentButtonShaper("org.jvnet.substance.button.StandardButtonShaper");

SubstanceLookAndFeel

.setCurrentWatermark("org.jvnet.substance.watermark.SubstanceBubblesWatermark");

SubstanceLookAndFeel

.setCurrentGradientPainter("SpecularGradientPainter");

try {

UIManager.setLookAndFeel(new SubstanceLookAndFeel());

} catch (UnsupportedLookAndFeelException e) {

e.printStackTrace();

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Chennal().setVisible(true);

}

});

}

byte[] imgb;

// GEN-BEGIN:variables

// Variables declaration - do not modify

private javax.swing.JLabel SName;

private javax.swing.JButton btnEncryption;

private javax.swing.JButton btnNeigh;

private javax.swing.JButton btnPathDiscovery;

private javax.swing.JButton btnSelectImage;

private javax.swing.JButton btnSend;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel9;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JTextArea jtaNeigh;

private javax.swing.JLabel lblImage;

public javax.swing.JLabel lblShortestPath;

private javax.swing.JLabel lblSource;

private javax.swing.JTable tblPaths;

private javax.swing.JTextField txtDestination;

private javax.swing.JTextField txtPrivateKey;

private javax.swing.JTextField txtPublicKey;

private javax.swing.JTextField txtSignature;

public DefaultTableModel dftQue;

// End of variables declaration//GEN-END:variables

}

**Action Class**

package com.dwt;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.Enumeration;

import java.util.Properties;

import java.util.Vector;

import javax.swing.JTextArea;

public class Action {

public String getVerificationCode(String path) {

// TODO Auto-generated method stub

String str = "";

try {

File file = new File(path);

str = file.getName();

str = str.substring(0, str.lastIndexOf("."));

str = VerificationCode.calculateCode(str, str);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return str;

}

public String getVerifiCode(String path) {

// TODO Auto-generated method stub

String str = "";

try {

str = VerificationCode.calculateCode(path, path + path);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return str;

}

public boolean getStatus() {

boolean flag = false;

try {

Properties p1 = new Properties();

FileInputStream fis = new FileInputStream("Attacker.properties");

p1.load(fis);

String sta = p1.getProperty("attack");

if (sta.equals("true"))

flag = true;

} catch (Exception ea) {

ea.printStackTrace();

}

return flag;

}

KeyPairs keys = new KeyPairs();

public String getSource() {

// TODO Auto-generated method stub

return new SourceAndPort().getSource();

}

public int getPort() {

// TODO Auto-generated method stub

return new SourceAndPort().getPort();

}

public void setProperty(String file, String source, String text) {

// TODO Auto-generated method stub

try {

Properties properties = new Properties();

FileOutputStream fos = new FileOutputStream(file, true);

properties.setProperty(source, text);

properties.store(fos, source);

fos.close();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

Vector<String> neigh = new Vector<String>();

public void setNeighbour(JTextArea jtaNeigh, String source) {

// TODO Auto-generated method stub

try {

Properties properties = new Properties();

FileInputStream fis = new FileInputStream("Distance.properties");

properties.load(fis);

fis.close();

int mydis = Integer.parseInt(properties.getProperty(source));

Enumeration<Object> em = properties.keys();

while (em.hasMoreElements()) {

String key = (String) em.nextElement();

int dis = Integer.parseInt(properties.getProperty(key));

int min = mydis - 50;

int max = mydis + 50;

if ((!key.equals(source)) && dis >= min && dis <= max) {

neigh.add(key);

jtaNeigh

.append(key + " ["

+ getProperty("Publickey.properties", key)

+ " ]\n");

}

}

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void routing(Vector<String> path, String dest) {

// TODO Auto-generated method stub

try {

for (int i = 0; i < neigh.size(); i++) {

String nei = neigh.get(i);

if (getAvailable(path, nei)) {

int nPort = getPort(nei);

Socket socket = new Socket("localhost", nPort);

ObjectOutputStream oos = new ObjectOutputStream(socket

.getOutputStream());

oos.writeObject("RTS");

oos.writeObject(path);

oos.writeObject(dest);

}

}

} catch (UnknownHostException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public String getProperty(String file, String key) {

// TODO Auto-generated method stub

String value = "";

try {

Properties properties = new Properties();

FileInputStream fis = new FileInputStream(file);

properties.load(fis);

fis.close();

value = properties.getProperty(key);

} catch (NumberFormatException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return value;

}

private boolean getAvailable(Vector<String> path, String nei) {

// TODO Auto-generated method stub

return !path.contains(nei);

}

public int getPort(String nei) {

// TODO Auto-generated method stub

int mydis = 0;

try {

Properties properties = new Properties();

FileInputStream fis = new FileInputStream("Ports.properties");

properties.load(fis);

fis.close();

mydis = Integer.parseInt(properties.getProperty(nei));

} catch (NumberFormatException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return mydis;

}

public void sendData(String dest, Data data) {

// TODO Auto-generated method stub

try {

int nPort = getPort(dest);

Socket socket = new Socket("localhost", nPort);

ObjectOutputStream oos = new ObjectOutputStream(socket

.getOutputStream());

oos.writeObject("Data");

oos.writeObject(data);

} catch (UnknownHostException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public String getPublicKey() {

// TODO Auto-generated method stub

return keys.getPublic();

}

public String getPrivateKey() {

// TODO Auto-generated method stub

return keys.getPrivate();

}

public int getDistance() {

// TODO Auto-generated method stub

return new SourceAndPort().getDistance();

}

}

**Verification Code:**

package com.dwt;

import java.security.SignatureException;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

public class VerificationCode {

private static final String HMAC\_SHA1\_ALGORITHM = "HmacSHA1";

public static String calculateCode(String data, String key)

throws java.security.SignatureException {

String result;

try { byte[] keyBytes = key.getBytes();

SecretKeySpec signingKey = new SecretKeySpec(keyBytes,

HMAC\_SHA1\_ALGORITHM);

Mac mac = Mac.getInstance(HMAC\_SHA1\_ALGORITHM);

mac.init(signingKey);

byte[] rawHmac = mac.doFinal(data.getBytes());

result = Base64Codec.encode(new String(rawHmac));

} catch (Exception e) {

throw new SignatureException("Failed to generate HMAC : "

+ e.getMessage()); }

return result; }

}

**7. TESTING**

**INTRODUCTION**

The process of executing program with the intent of finding errors in them is called testing. During testing, the program to be tested is executed with a set of test cases, and the output of the program for the test cases is evaluated to determine if programs is performing as expected. Testing forms the first step in determining the errors in the program. The success of testing in revealing errors in program depends critically on test cases.

**Functional Testing:**

There are two basic approaches of functional testing:

a. Black box or functional testing.

b. White box testing or structural testing.

**7.1 Black box testing**

This method is used when knowledge of the specified function that a product has been design to perform is known. The concept of black box is used to represent a system hose inside working’s are not available to inspection. In a black box the test item is eaten as “Black”, since its logic is unknown is what goes in and what comes out, or the input and output.

In **black box testing,** we try various inputs and examine the resulting outputs. The black box testing can also be used for scenarios based test .In this test we verify whether it is taking valid input and producing resultant out to user. It is imaginary box testing that hides internal workings. In our project valid input is image resultant output well structured image should be received.

Input output

EXTERNAL WORKING

* 1. **White box testing**

White box testing is concern with testing implementation of the program. The intent of structural testing is not to exercise all the inputs or outputs but to exercise the different programming and data structures used in the program. Thus structure testing aims to achieve test cases that will force the desire coverage of different structures. Two types of path testing are:

1. Statement testing

2. Branch testing

**Statement Testing**

The main idea of statement testing coverage is to test every statement in the objects method by executing it at least once. However, realistically, it is impossible to test program on every single input, so you never can be sure that a program will not fail on some input.

**Branch Testing**

The main idea behind branch testing coverage is to perform enough tests to ensure that every branch alternative has been executed at least once under some test. As in statement testing coverage, it is unfeasible to fully test any program of considerable size.

InputOutput

INTERNAL

WORKING

**7.3 Unit Testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**7.4 Integration Testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**7.5 Functional Testing**

Functional tests provide a systematic demonstration that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify business process flows, data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

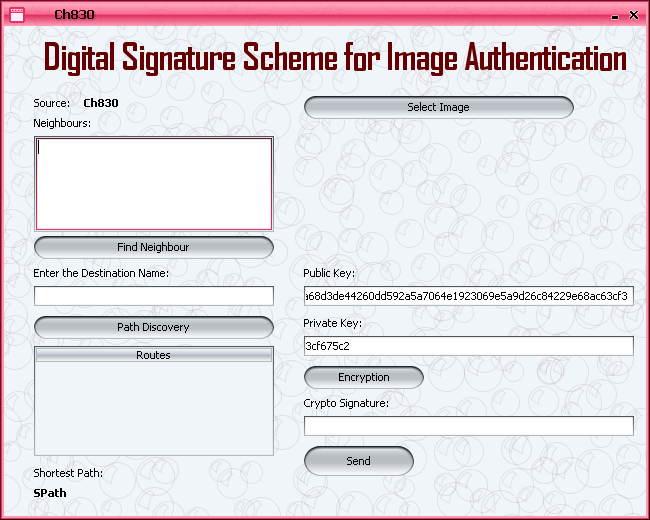
**7.6 System Testing**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

1. **OUTPUT SCREENS**

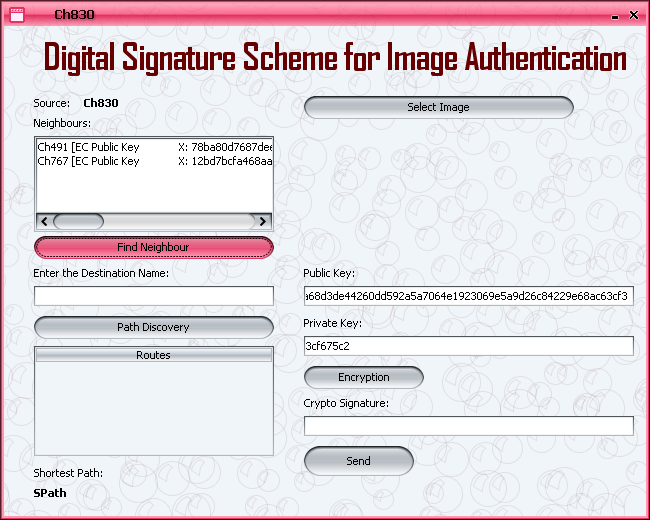
**Screen 1**

It is channel of sender side. It consist of different buttons such as Find Neighbour, Path discovery, select image and send.



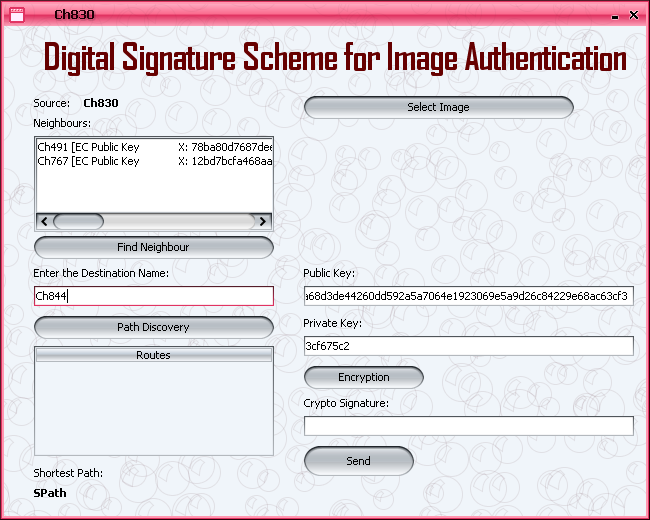
**Screen2**

Here we identify the neighbour channels based on those channels we identify the destination to send image.



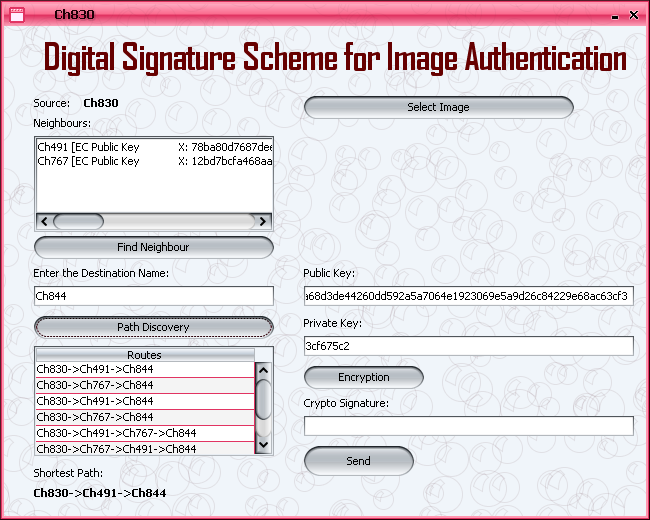
**Screen3**

Here we enter destination to send the image securely.



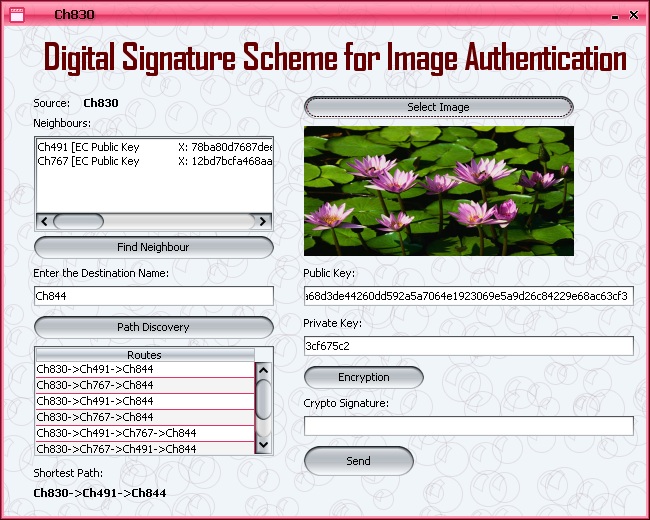
**Screen4**

Path Discovery button will display paths to receiver. Shortest path discover shortest path to send image.



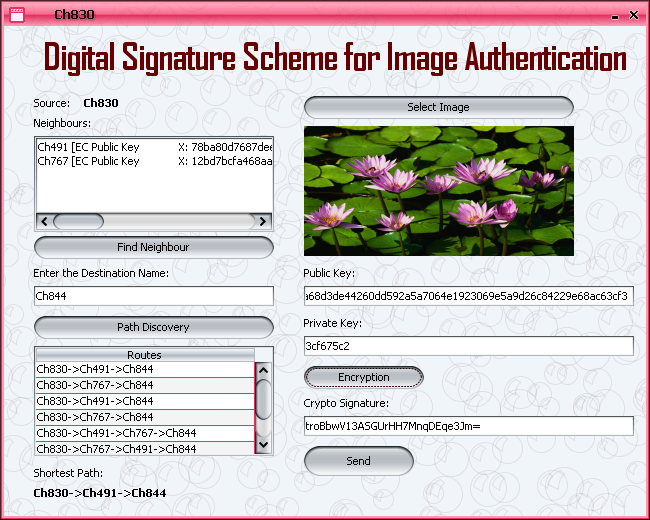
**Screen5**

Here, we select image to send receiver.



**Screen6**

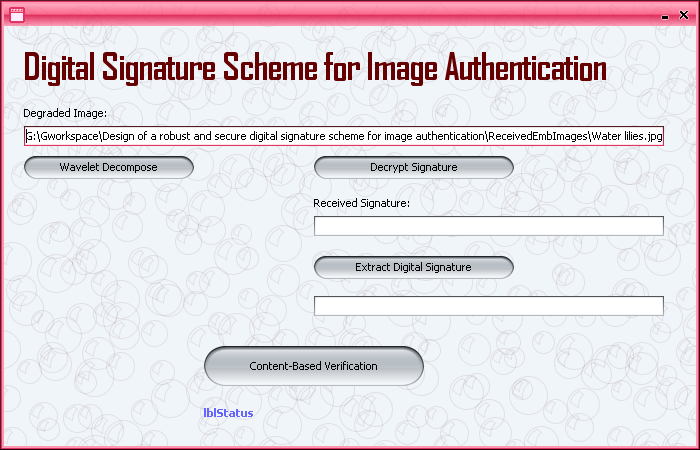
Here, we generate crypto signature by clicking on encryption. Finally we send the image to receiver then and image received message box is displayed.





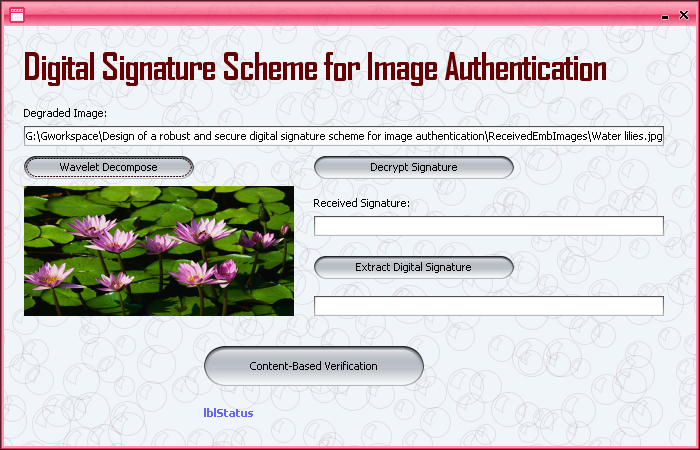
**Screen7**

By clicking on Ok in message box we generate a channel i.e., receiver channel.



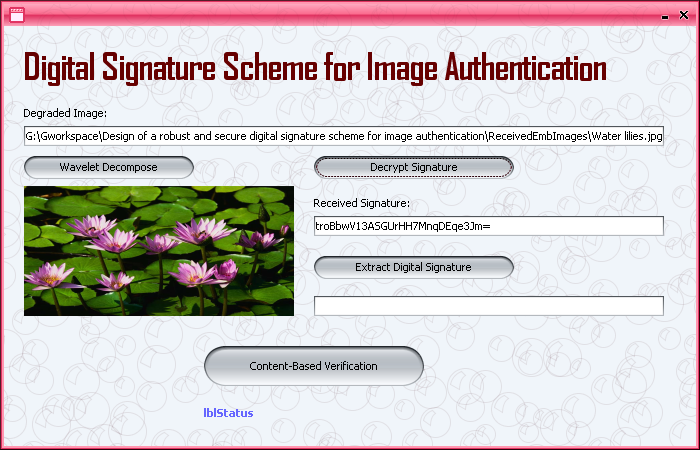
**Screen8**

By clicking on wavelet decomposition we generate image.



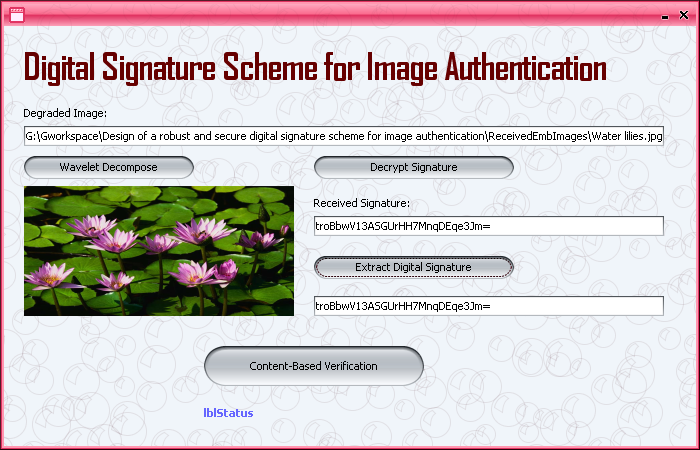
**Screen9**

By clicking on decrypt signature we generate received signature which is encrypted by sender.



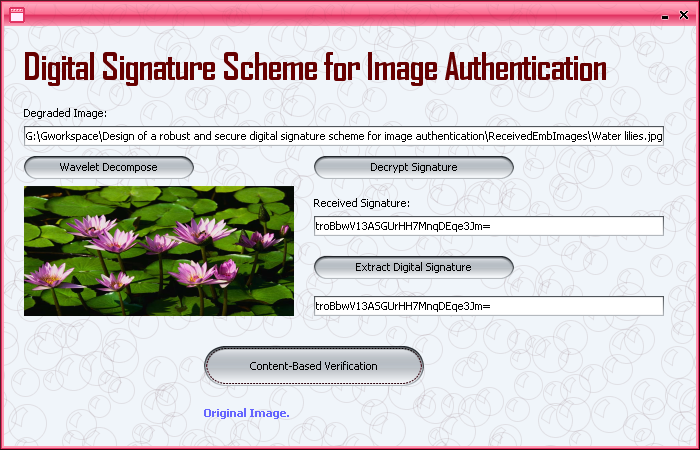
**Screen10**

By clicking on Extract digital signature we generate a signature from image.



**Screen11**

If both signatures i.e., received signature and extracted signature are equal then it is original image.



**Screen12**

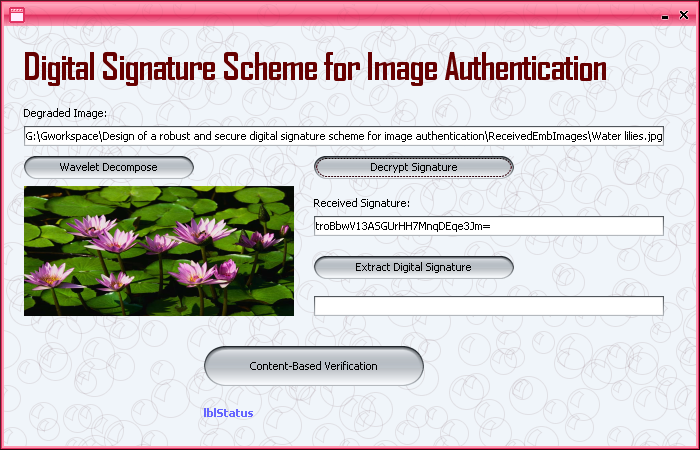
Keep attacker in ready state to attack image while transferring and send image.

With Attacker:



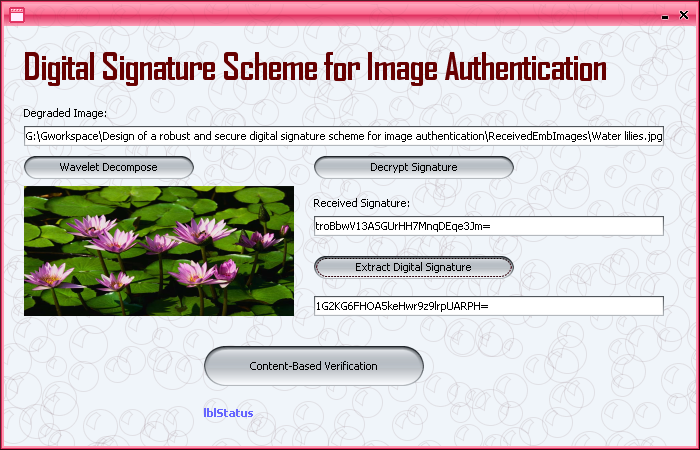
**Screen13**

Here also we generate a signature which is changed compared to previous, because attacker attacked image in middle. Instead to verify it we generate a digital signature from image also.



**Screen14**

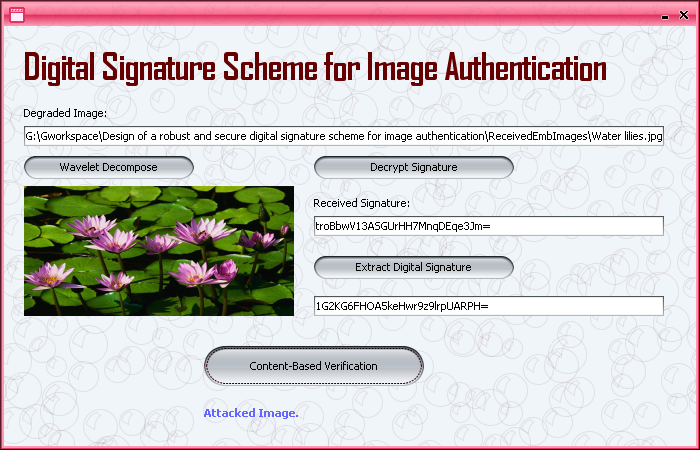
Here also we generate a digital signature from image and compared with received signature. Here both are not equal so it will generate a message box Message attacked.





**Screen15**

Here it intimates receiver that message is attacked at receiver side.



**9. CONCLUSION**

A modified digital signature scheme for image authentication has been proposed. Content-dependent structural image features and wavelet filter parameterization are incorporated into the traditional crypto signature scheme to enhance the system robustness and security. Because the proposed scheme does not require any computational overhead, it is especially suited for wireless authentication systems and other real-time applications.

Content-dependent structural image features and wavelet, It parameterization are incorporated into the traditional crypto signature scheme to enhance the system robustness and security. Because the proposed scheme does not require any computational overhead, it is especially suited for wireless authentication systems and other real-time applications.

1. **Further Enhancements**

The Image authentication over wireless channels software can be enhanced in the future in different kinds. The analysis and the experimental results confirm that the proposed scheme can achieve good robustness against transmission errors and some acceptable manipulation operations.

The scheme is very robust to cutting and pasting counterfeiting attacks. It is also able to tolerate various common image processing manipulations, at the cost of only extra payload introduced into the channel by associating the signature with the image. Further work will conduct more tests on the quality of degraded images.

**11. REFERENCES**

Under this references section, we have mentioned various references from which we collected our problem and several others that supported us to design the solution for our problem. These references include either books, papers published through some standards and several websites links with URL’s:

* Department of information Technology, Institute of Graduate Studies and research, Universities of Alexandria,163 Horreya Avenue EI shatby, po Box 832,Alexandria,Egypt . E-mail:saad.darwish@gmail.com
* GINESU G., GIUSTO D.D., ONALI T.: ‘Mutual image based authentication framework with JPEG2000 in wireless environment’, EURASIP J. Wirel. Commun. Netw., 2006, 2006, pp. 1–14 (Article ID 73685) .
* Java Complete Reference,TMH.
* SUN Q., YE S., LIN C.-Y.: ‘A crypto signature scheme for image authentication over wireless channel’, Int. J. Image Graph., 2005, 5, (1), pp. 1–14.

**12. APPENDICES**

# Appendix A: Glossary

TERMS

          All the terms and abbreviation in the project are specified clearly. For further development of project evolved definitions will be specified

ACRONYMS

         IEEE:   Institute of Electrical and Electronics Engineers

DFD: Data Flow Diagram

J2EE:Java2 Enterprise Edition

GUI:Graphical User Interface

SDS: Structural Digital Signature

JPEG: Joint photographic Experts Group

**Appendix B: Analysis Models**

This includes all the pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.

**Appendix C: To Be Determined List**