**CHAPTER 1: INTRODUCTION**

**1.1 Vision**

In this technology, the end user identifies an image, which is going to act as the carrier of data. The data file is also selected and then to achieve greater speed of transmission the data file and image file are compressed and sent. Prior to this the data is embedded into the image and then sent. The image if hacked or interpreted by a third party user will open up in any image previewed but not displaying the data. This protects the data from being invisible and hence is secure during transmission. The user in the receiving end uses another piece of code to retrieve the data from the image.

**1.2 SCOPE**

System provides a friendly environment to deal with images. Generally tools supports only one kind of image formats. Our application supports .gif and .png (portable network graphics formatted images and our application has been developed using swing and applet technologies hence provides a friendly environment to users.

**CHAPTER 2: PROBLEM DESCRIPTION AND REQUIREMENT ANALYSIS**

**2.1 Existing System:**

As it is a known fact that Indian Railways has the largest railway network in the world and everyday millions of railway tickets are booked. They still use the command based approach for ticket booking, which is now considered as an outdated approach. Considering the number of booking counters available and the long queues at the reservation counter, it takes a lot of time for the passenger to get a ticket reserved or canceled. The passenger has to fill the reservation form with the train details such as train number; train name, date of journey, boarding and destination station and details of passengers who are travelling. Thus to fill the reservation form it is a difficult task for the passenger to find out the train details as there are thousands of trains.

**2.2 Proposed System:**

The goal of online railway reservation is easing the tedious task of railway activity. All the required information about the trains such as different trains, availability of seats, schedules of a particular train, fare list for different trains all of the above details would be available at one place for any user. The passenger who wishes to make reservation and cancellation of tickets online through the website has to register first. The registered user after login validation can easily make booking for available trains on his desired date of journey and cancellation of seats which he had booked earlier.

**CHAPTER 3 : SYSTEM REQUIREMENTS SPECIFICATION**

**3.1 Software Requirements**

Operating System:  Windows   X

Technologies: JAVA 6.0, Swing

Tools: NetBeans 6.0

**3.2 Hardware Requirements**

**Requirements for the Remote Machine Environment:**

Processor: Intel or AMD processor computer

RAM: 256 MB or more

Hard Disk Space: 8 GB or more

**3.3 OTHER** **REQUIREMENTS**

**3.3.1 Functional Requirements**

Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified.

**3.3.2 Non-Functional Requirements**

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy ((.e. how precise are the systems numerical answers.)

**3.3.3 Pseudo Requirements**

The client that restricts the implementation of the system imposes these requirements. Typical pseudo requirements are the implementation language and the platform on which the system is to be implemented. These have usually no direct effect on the users view of the system**.**

**CHAPTER 4 : SOURCE CODE**

**Anon.util**

**Classutil.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package jb;

import javax.swing.JApplet;

import javax.swing.JFrame;

/\*\* run the VCApplet as an application

\*

\*/

public class VCApplication{

static final JApplet applet = new VCApplet();

/\*\*starts the applet in a frame

\*

\* @param argv - the normal argv[] - here it is empty

\*/

static public void main (String argv[]) {

// if(argv.length==1){

// JFrame frame = new JFrame ("Embedded Extended Visual Cryptography Schemes");

// frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// frame.getContentPane().add ("Center", applet);

//// applet.setStub(new VCAppletStub (applet, argv));

// frame.setSize(700,700);

// frame.setVisible(true);

// applet.init();

// applet.start();

//// frame.pack();

// }else{

// //print out the usage term

// System.out.println("usage: java VCApplication host=<path>");

// System.out.println("<path> = path to the pictures to load");

// System.out.println("for windows for example:\n host=\"C:\\Dokumente und Einstellungen\\Boßle Johannes\\Eigene Dateien\\sec\\vc\_work\\vc\"");

// System.out.println("or \"\" if nothing was changed (vc.jar and the pics are in the same directory)");

// System.out.println("trying to continue anyway...");

// String[] arg = {"host=\"\""};

// VCApplication.main(arg);

// }

JFrame frame = new JFrame ("Embedded Extended Visual Cryptography Schemes");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().add ("Center", applet);

// applet.setStub(new VCAppletStub (applet, argv));

frame.setSize(700,740);

frame.setVisible(true);

applet.init();

applet.start();

}

}

**ResourceInstantiator.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

JP iNFOTeCH

\*/

package anon.util;

import java.io.File;

import java.util.zip.ZipEntry;

import java.util.zip.ZipFile;

/\*\*

\* This class is only needed in the context of the ResourceLoader. It loads resources

\* and transforms them into concrete objects.

\* @author Rolf Wendolsky

\* @see anon.util.ResourceLoader

\*/

public interface ResourceInstantiator

{

/\*\*

\* Loads a file and transforms it into a concrete object.

\* @param a\_file a File

\* @param a\_topDirectory the top directory of this file; this is generally not needed to load

\* the file

\* @throws Exception if an error occurs

\* @return an Object

\*/

public Object getInstance(File a\_file, File a\_topDirectory) throws Exception;

/\*\*

\* Loads a file and transforms it into a concrete object. Please not that JDK 1.1.8 does not

\* correctly load zip entries! Please use the method ResourceLoader.loadResource(String)

\* instead. By this way, only zip/jar files in the classpath may be loaded.

\* @param a\_entry a ZipEntry

\* @param a\_file the ZipFile of this zip entry

\* @throws Exception if an error occurs

\* @return an Object

\*/

public Object getInstance(ZipEntry a\_entry, ZipFile a\_file) throws Exception;

/\*\*

\* An Exception that is thrown by a ResourceInstantiator if too much instantiations have failed.

\* The instantiator object will be invalid after throwing this exception.

\* @author Rolf Wendolsky

\*/

public class ResourceInstantiationException extends Exception

{

/\*\*

\*

\*/

private static final long serialVersionUID = 5654908530903321864L;

}

}

**ResourceLoader.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

JP iNFOTeCH

\*/

package anon.util;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.File;

import java.net.URL;

import java.net.MalformedURLException;

import java.util.Vector;

import java.util.Hashtable;

import java.util.Enumeration;

import java.util.StringTokenizer;

import java.util.zip.ZipEntry;

import java.util.zip.ZipFile;

/\*\*

\* This class loads resources from the file system. It allows to specify resource paths

\* like "dir1/dir2/resource" or "dir1/dir2/" relative to the classpath or the current directory.

\* It allows also for going up directories like "dir1/dir2/../resource" that would be translated

\* to "dir1/resource".

\* @author Rolf Wendolsky

\*/

public final class ResourceLoader

{

private static final String SYSTEM\_RESOURCE\_TYPE\_ZIP = "ZIP";

private static final String SYSTEM\_RESOURCE\_TYPE\_JAR = "JAR";

private static final String SYSTEM\_RESOURCE\_TYPE\_FILE = "FILE";

private static final String SYSTEM\_RESOURCE = "systemresource:/";

private static final String SYSTEM\_RESOURCE\_ENDSIGN = "/+/";

private static final String DIR\_UP = "../";

// private static final String DIR\_CURRENT = "./";

private static final int READ\_BUFFER = 2000;

private static final String RESOURCE\_NO\_CLASSES\_FOUND = "";

/// holds references to all files in the class path as File objects for caching purposes

private static Vector ms\_classpathFiles;

/// holds absolute URLs to the resources as Strings; relative paths may be attached

private static Vector ms\_classpathResourceURLs = new Vector();

/// the resource types: either SYSTEM\_RESOURCE\_TYPE\_ZIP or SYSTEM\_RESOURCE\_TYPE\_FILE

private static Vector ms\_classpathResourceTypes;

/// stores the parent directory of jar file that holds this class for caching purposes

private static File ms\_parentResourceFile;

private static String ms\_parentResourceFileResourceURL;

private static String ms\_parentResourceFileResourceType;

/// the class path at the last state it was read

private static String ms\_classpath;

{

try

{

ms\_parentResourceFile =

new File(ClassUtil.getClassDirectory(

ResourceLoader.class).getAbsolutePath());

}

catch (Exception a\_e)

{

// the parent resource file could not be loaded

a\_e.printStackTrace(System.err);

ms\_parentResourceFile = null;

}

}

private ResourceLoader()

{

}

/\*\*

\* Reads a java.io.InputStream into a byte array and closes the stream.

\* If the stream blocks before the first byte is read this method will block, too, until

\* there are any bytes available for reading.

\* @param a\_iStream an InputStream

\* @throws IOException if an I/O error occurs

\* @return the InputStream as bytes

\*/

public static byte[] getStreamAsBytes(InputStream a\_iStream) throws IOException

{

byte[] data;

byte[] buffer;

int readDataLength = 1;

data = new byte[0];

// if readDataLength >= 0 there are more bytes available to read

while(readDataLength >= 0)

{

// initialize the buffer

if (a\_iStream.available() > 0)

{

buffer = new byte[a\_iStream.available()];

}

else

{

buffer = new byte[READ\_BUFFER];

}

// read all available data into buffer until the buffer is filled

readDataLength = a\_iStream.read(buffer);

data = trimByteArray(buffer, readDataLength, data);

}

a\_iStream.close();

return data;

}

/\*\*

\* Gets the absolute URL to a requested resource if the resource is found in the class path or

\* in the local directory. Loads a single resource only, therefore directory specifications like

\* "home/dir/" are not allowed.

\* @param a\_strRelativeResourcePath the relative path to a resource

\* @return the absolute URL to the requested resource

\*/

public static URL getResourceURL(String a\_strRelativeResourcePath)

{

Vector parentResourceFile;

Vector parentResourceFileURL;

Vector parentResourceFileType;

File localFile;

URL resourceURL = null;

if ((a\_strRelativeResourcePath = formatResourcePath(a\_strRelativeResourcePath)) == null ||

a\_strRelativeResourcePath.endsWith("/"))

{

return null;

}

// this is the standard method for getting a resource URL from the class path

resourceURL = ResourceLoader.class.getResource("/" + a\_strRelativeResourcePath);

if (resourceURL == null)

{

// try to find the resource in the local directory

localFile = new File(a\_strRelativeResourcePath);

if (localFile.exists() && localFile.canRead())

{

try

{

resourceURL = new java.net.URL("file:" + localFile.getAbsolutePath());

}

catch (MalformedURLException a\_e)

{

}

}

}

if (resourceURL == null && ms\_parentResourceFile != null &&

!readFilesFromClasspath().contains(ms\_parentResourceFile))

{

/\*\*

\* The parent resource file is not contained in the class path.

\* Try to load the requested resource directly from the local file.

\*/

parentResourceFile = new Vector();

parentResourceFileURL = new Vector();

parentResourceFileType = new Vector();

parentResourceFile.addElement(ms\_parentResourceFile);

parentResourceFileURL.addElement(ms\_parentResourceFileResourceURL);

parentResourceFileType.addElement(ms\_parentResourceFileResourceType);

resourceURL = getResourceURL(a\_strRelativeResourcePath, parentResourceFile,

parentResourceFileURL, parentResourceFileType);

ms\_parentResourceFileResourceURL = (String) parentResourceFileURL.firstElement();

ms\_parentResourceFileResourceType = (String) parentResourceFileType.firstElement();

}

/\*\*

\* This is an other (tricky) implementation that tries to find the resource in the

\* classpath. It may be used if class.getResource(..) does not work properly.

\* Please do not remove as this could be important for testing purposes.

if (resourceURL == null)

{

// classPathFiles and classPathResources must be synchronized!

synchronized (ms\_classpathResourceURLs)

{

resourceURL = getResourceURL(a\_strRelativeResourcePath, readFilesFromClasspath(),

ms\_classpathResourceURLs, ms\_classpathResourceTypes);

}

}\*/

return resourceURL;

}

/\*\*

\* Loads a resource from the classpath or the current directory.

\* The resource may be contained in an archive (ZIP,JAR) or a directory structure.

\* If the resource could not be found in the classpath, it is loaded from the current

\* directory. Loads a single resource only, therefore directory specifications like

\* "home/dir/" are not allowed.

\* @param a\_strRelativeResourcePath a relative filename for the resource

\* @return the contents of the resource or null if resource could not be loaded

\*/

public static byte[] loadResource(String a\_strRelativeResourcePath)

{

InputStream in;

if ((a\_strRelativeResourcePath = formatResourcePath(a\_strRelativeResourcePath)) == null ||

a\_strRelativeResourcePath.endsWith("/"))

{

return null;

}

// load images from the local classpath

in = ResourceLoader.class.getResourceAsStream("/" + a\_strRelativeResourcePath);

try

{

if (in == null && ms\_parentResourceFile != null &&

!readFilesFromClasspath().contains(ms\_parentResourceFile))

{

/\*\*

\* The parent resource file is not contained in the class path. Try to load the

\* resource directly from the parent resource file.

\*/

in = new FileInputStream(

new File(ms\_parentResourceFile,

replaceFileSeparatorsSystemSpecific(a\_strRelativeResourcePath)));

}

}

catch (IOException a\_e)

{

}

try

{

if (in == null)

{

// load resource from the current directory

in = new FileInputStream(a\_strRelativeResourcePath);

}

return getStreamAsBytes(in);

}

catch (IOException a\_e)

{

return null;

}

}

/\*\*

\* Loads resources from the classpath or the current directory and instantiates them as

\* byte arrays. Resources with the same path name

\* are only loaded from the first source that contains the resource.

\* If the resource search path is a directory, the method returns a Hashtable with all

\* instanciated resources. Otherwise, the Hashtable contains only one element or is empty if

\* no resource was found or no resource could not be instantiated.

\* The resources may be contained in an archive (JAR) or a directory structure. If no resources

\* could be found in the classpath or they could not be instanciated by the given

\* resource instantiator, they are loaded from the current directory.

\* The resource search path is relative to the given directory and may either specify a single

\* resource or, if it ends with a slash "/", a (virtual) directory in the given directory.

\* Therefore a resource search path "/" loads all resources in the specified directory.

\* A resource search path "certificates/" loads all resources in the (virtual) directory

\* <i> certificates </i>.

\* @param a\_strResourceSearchPath a relative filename for the resource

\* @param a\_bRecursive true if (virtual) subdirectories should be searched for resources;

\* false otherwise (has an effect only for resource paths ending with "/")

\* @return the contents of the resources as byte arrays or an empty Hashtable if no resource

\* could be loaded

\*/

public static Hashtable loadResources(String a\_strResourceSearchPath, boolean a\_bRecursive)

{

return loadResources(a\_strResourceSearchPath,

(new ResourceLoader()).createByteArrayInstantiator(),

a\_bRecursive);

}

/\*\*

\* Loads resources from the classpath or the current directory and instantiates them as

\* objects of a type defined by the resource instantiator. Resources with the same path name

\* are only loaded from the first source that contains the resource.

\* If the resource search path is a directory, the method returns a Hashtable with all

\* instanciated resources. Otherwise, the Hashtable contains only one element or is empty if

\* no resource was found or no resource could not be instantiated.

\* The resources may be contained in an archive (JAR) or a directory structure. If no resources

\* could be found in the classpath or they could not be instanciated by the given

\* resource instantiator, they are loaded from the current directory.

\* The resource search path is relative to the given directory and may either specify a single

\* resource or, if it ends with a slash "/", a (virtual) directory in the given directory.

\* Therefore a resource search path "/" loads all resources in the specified directory.

\* A resource search path "certificates/" loads all resources in the (virtual) directory

\* <i> certificates </i>.

\* @param a\_strResourceSearchPath a relative filename for the resource

\* @param a\_instantiator an object that instantiates the loaded resource

\* @param a\_bRecursive true if (virtual) subdirectories should be searched for resources;

\* false otherwise (has an effect only for resource paths ending with "/")

\* @return the contents of the resource or an empty Hashtable if no resource

\* could be loaded

\*/

public static Hashtable loadResources(String a\_strResourceSearchPath,

ResourceInstantiator a\_instantiator,

boolean a\_bRecursive)

{

Hashtable resources = new Hashtable();

Enumeration classPathFiles = readFilesFromClasspath().elements();

while (classPathFiles.hasMoreElements())

{

loadResources(a\_strResourceSearchPath, (File)classPathFiles.nextElement(),

a\_instantiator, a\_bRecursive, false, resources);

}

loadResources(a\_strResourceSearchPath, new File(System.getProperty("user.dir")),

a\_instantiator, a\_bRecursive, false, resources);

return resources;

}

/\*\*

\* Loads resources from a directory and instantiates them as byte arrays.

\* If the resource search path is a directory, the method returns a Hashtable with all

\* instanciated resources. Otherwise, the Hashtable contains only one element or is empty if

\* no resource was found or no resource could not be instantiated.

\* The resources may be contained in an archive (JAR) or a directory structure.

\* The resource search path is relative to the given directory and may either specify a single

\* resource or, if it ends with a slash "/", a (virtual) directory in the given directory.

\* Therefore a resource search path "/" loads all resources in the specified directory.

\* A resource search path "certificates/" loads all resources in the (virtual) directory

\* <i> certificates </i>.

\* @param a\_strResourceSearchPath a relative filename for the resource

\* @param a\_directory a simple file, directory or zip file

\* @param a\_bRecursive true if (virtual) subdirectories should be searched for resources;

\* false otherwise (has an effect only for resource paths ending with "/")

\* @return the contents of the resource or an empty Hashtable if no resource

\* could be loaded

\*/

public static Hashtable loadResources(String a\_strResourceSearchPath,

File a\_directory,

boolean a\_bRecursive)

{

Hashtable resources = new Hashtable();

loadResources(a\_strResourceSearchPath, a\_directory,

(new ResourceLoader()).createByteArrayInstantiator(),

a\_bRecursive, false, resources);

return resources;

}

/\*\*

\* Loads resources from a directory and instantiates them as

\* objects of a type defined by the resource instantiator.

\* If the resource search path is a directory, the method returns a Hashtable with all

\* instanciated resources. Otherwise, the Hashtable contains only one element or is empty if

\* no resource was found or no resource could not be instantiated.

\* The resources may be contained in an archive (JAR) or a directory structure.

\* The resource search path is relative to the given directory and may either specify a single

\* resource or, if it ends with a slash "/", a (virtual) directory in the given directory.

\* Therefore a resource search path "/" loads all resources in the specified directory.

\* A resource search path "certificates/" loads all resources in the (virtual) directory

\* <i> certificates </i>.

\* @param a\_strResourceSearchPath a relative filename for the resource

\* @param a\_directory a simple file, directory or zip file

\* @param a\_instantiator an object that instantiates the loaded resource

\* @param a\_bRecursive true if (virtual) subdirectories should be searched for resources;

\* false otherwise (has an effect only for resource paths ending with "/")

\* @return the contents of the resource or an empty Hashtable if no resource

\* could be loaded

\*/

public static Hashtable loadResources(String a\_strResourceSearchPath,

File a\_directory,

ResourceInstantiator a\_instantiator,

boolean a\_bRecursive)

{

Hashtable resources = new Hashtable();

loadResources(a\_strResourceSearchPath, a\_directory,

a\_instantiator, a\_bRecursive, false, resources);

return resources;

}

/\*\*

\* Interprets a String as a filename and converts its file separators to

\* system specific file separators.

\* @param a\_filename a generic file name

\* @return a system specific file name

\*/

public static String replaceFileSeparatorsSystemSpecific(String a\_filename)

{

if (a\_filename == null)

{

return null;

}

a\_filename = a\_filename.replace('/', File.separatorChar);

a\_filename = a\_filename.replace('\\', File.separatorChar);

return a\_filename;

}

/\*\*

\* Returns a given a requested system resource as a file. A system resource is either

\* a zip file or a directory that is specified in the classpath

\* (Property <Code> java.class.path </Code>). The resource must be specified with by

\* the following protocol syntax:

\* <DL>

\* <DT> ZipFile </DT>

\* <DD> :systemresource:/ZIP[id]/+/ </DD>

\* <DT> File </DT>

\* <DD> :systemresource:/FILE[id]/+/ </DD>

\* </DL>

\* [id] may be an integer specifying the resource's position int the classpath

\* (beginning with 0) or an absolute path containing the requested resource. The end sign

\* '/+/' is optional and marks the end of the [id].

\* The system resource protocol is only used in old JDKs < 1.2.

\*

\* @param a\_systemResource a system resource a String

\* @return The requested system resource as a file or null if the resource could not be found

\*/

protected static File getSystemResource(String a\_systemResource)

{

int endIndex;

if (a\_systemResource.indexOf(SYSTEM\_RESOURCE) != 0)

{

return null;

}

// find the beginning of the [id] string

a\_systemResource =

a\_systemResource.substring(SYSTEM\_RESOURCE.length(), a\_systemResource.length());

if (a\_systemResource.toUpperCase().startsWith(SYSTEM\_RESOURCE\_TYPE\_ZIP))

{

a\_systemResource = a\_systemResource.substring(

SYSTEM\_RESOURCE\_TYPE\_ZIP.length(), a\_systemResource.length());

}

else if (a\_systemResource.toUpperCase().startsWith(SYSTEM\_RESOURCE\_TYPE\_JAR))

{

a\_systemResource = a\_systemResource.substring(

SYSTEM\_RESOURCE\_TYPE\_JAR.length(), a\_systemResource.length());

}

else if (a\_systemResource.toUpperCase().startsWith(SYSTEM\_RESOURCE\_TYPE\_FILE))

{

a\_systemResource = a\_systemResource.substring(

SYSTEM\_RESOURCE\_TYPE\_FILE.length(), a\_systemResource.length());

}

// now find the end of the [id] string and extract the [id]

endIndex = a\_systemResource.indexOf(SYSTEM\_RESOURCE\_ENDSIGN);

if (endIndex >= 0)

{

a\_systemResource = a\_systemResource.substring(0, endIndex);

}

// try to interpret the [id] as an integer number

try

{

return (File)readFilesFromClasspath().elementAt(Integer.parseInt(a\_systemResource));

}

catch (Exception a\_e)

{

// the [id] seems to be a file path

return new File(a\_systemResource);

}

}

/\*\*

\* Loads resources from a given directory (simple file, directory or zip file) that are found

\* in the given resource search path and instantiates them as objects of a type defined by the

\* resource instantiator. Resources with the same path name

\* are only loaded once according to the given Hashtable.

\* The resource search path is relative to the given directory and may either specify a single

\* resource or, if it ends with a slash "/", a (virtual) directory in the given directory.

\* Therefore a resource search path "/" loads all resources in the specified directory.

\* A resource search path "certificates/" loads all resources in the (virtual) directory

\* <i> certificates </i>.

\* @param a\_strResourceSearchPath a (virtual) path in the directory to load resources from

\* @param a\_Directory a simple file, directory or zip file

\* @param a\_instantiator a ResourceInstantiator that is used to instantiate the loaded resources

\* @param a\_bRecursive true if (virtual) subdirectories should be searched for resources;

\* false otherwise (has an effect only for resource paths ending with "/")

\* @param a\_bStopAtFirstResource true if the search should stop with the first loaded resource;

\* false otherwise

\* @param a\_loadedResources a Hashtable where the loaded and instantiated resources are stored

\*/

protected static void loadResources(String a\_strResourceSearchPath,

File a\_Directory,

ResourceInstantiator a\_instantiator,

boolean a\_bRecursive,

boolean a\_bStopAtFirstResource,

Hashtable a\_loadedResources)

{

Enumeration entries;

if ((a\_strResourceSearchPath = formatResourcePath(a\_strResourceSearchPath)) == null ||

a\_loadedResources == null || a\_Directory == null || a\_instantiator == null ||

!a\_Directory.exists() || !a\_Directory.canRead())

{

return;

}

try

{

// try to fetch the objects as the file was a zip file

if (a\_Directory.isDirectory())

{

throw new IOException("This is a directory.");

}

Object object;

ZipFile zipfile;

ZipEntry zipentry;

String strCurrentResourcePath;

zipfile = new ZipFile(a\_Directory);

// if the search path is a single file, try to load it directly

if (!a\_strResourceSearchPath.endsWith("/"))

{

zipentry = zipfile.getEntry(a\_strResourceSearchPath);

if (zipentry == null)

{

throw new IOException ("Requested entry not found.");

}

// we have found the requested entry

Vector temp = new Vector();

temp.addElement(zipentry);

entries = temp.elements();

}

else

{

// search all entries for the requested ones

entries = zipfile.entries();

}

while (entries.hasMoreElements())

{

zipentry = (ZipEntry) entries.nextElement();

if (zipentry.isDirectory() ||

!isResourceInSearchPath(zipentry.toString(), a\_strResourceSearchPath,

a\_bRecursive))

{

continue;

}

object = null;

try

{

object = a\_instantiator.getInstance(zipentry, zipfile);

}

catch (ResourceInstantiator.ResourceInstantiationException a\_e)

{

return;

}

catch (Exception a\_e)

{

}

if (object != null)

{

strCurrentResourcePath = getCurrentResourcePath(zipentry);

if (!a\_loadedResources.containsKey(strCurrentResourcePath))

{

a\_loadedResources.put(strCurrentResourcePath, object);

if (!a\_strResourceSearchPath.endsWith("/") || a\_bStopAtFirstResource)

{

// the requested resource has been found

return;

}

}

}

}

}

catch (Exception a\_e)

{

// this seems to be no valid zip file; treat it as a simple file or directory

try

{

loadResourcesFromFile(a\_strResourceSearchPath, a\_Directory, a\_Directory,

a\_instantiator, a\_loadedResources,

a\_bRecursive, a\_bStopAtFirstResource);

}

catch (ResourceInstantiator.ResourceInstantiationException a\_ex)

{

return;

}

}

}

/\*\*

\* Returns all resources in a directory and the directory itself as Objects

\* if they are resources of the type specified by the ResourceInstantiator.

\* @param a\_strResourceSearchPath the (virtual) path in the file to load resources from

\* @param a\_file a resource file or directory

\* @param a\_topDirectory the directory where all other files and directories reside

\* @param a\_instantiator an object that instantiates the loaded resource

\* @param a\_loadedResources a Vector where the loaded and instantiated resources are stored

\* @param a\_bRecursive true if subdirectories should be visited; false otherwise

\* @param a\_bStopAtFirstResource true if the search should stop with the first loaded resource;

\* false otherwise

\* @throws ResourceInstantiator.ResourceInstantiationException if the ResourceInstantiator

\* has become invalid because of too many errors

\*/

private static void loadResourcesFromFile(String a\_strResourceSearchPath,

File a\_file, File a\_topDirectory,

ResourceInstantiator a\_instantiator,

Hashtable a\_loadedResources,

boolean a\_bRecursive,

boolean a\_bStopAtFirstResource)

throws ResourceInstantiator.ResourceInstantiationException

{

String[] filesArray;

String strCurrentResourcePath;

if ((!a\_strResourceSearchPath.endsWith("/") || a\_bStopAtFirstResource)

&& a\_loadedResources.size() > 0)

{

// the requested resource has already been found

return;

}

if (a\_file != null && a\_file.exists())

{

strCurrentResourcePath = getCurrentResourcePath(a\_file, a\_topDirectory);

// jump to the search path if the current file is not contained in the search path

if (strCurrentResourcePath.indexOf(a\_strResourceSearchPath) != 0 &&

!a\_strResourceSearchPath.equals("/"))

{

a\_file = new File(a\_topDirectory,

replaceFileSeparatorsSystemSpecific(a\_strResourceSearchPath));

loadResourcesFromFile(a\_strResourceSearchPath, a\_file, a\_topDirectory,

a\_instantiator, a\_loadedResources,

a\_bRecursive, a\_bStopAtFirstResource);

return;

}

if (a\_file.isFile() && isResourceInSearchPath(

strCurrentResourcePath, a\_strResourceSearchPath, a\_bRecursive))

{

Object object = null;

if (a\_loadedResources.containsKey(strCurrentResourcePath))

{

// this file has already been loaded

return;

}

try

{

object = a\_instantiator.getInstance(a\_file, a\_topDirectory);

}

catch (ResourceInstantiator.ResourceInstantiationException a\_e)

{

throw a\_e;

}

catch (Exception a\_e)

{

}

if (object != null)

{

a\_loadedResources.put(strCurrentResourcePath, object);

if (!a\_strResourceSearchPath.endsWith("/") || a\_bStopAtFirstResource)

{

// the requested resource has been found

return;

}

}

}

else if (a\_file.isDirectory() && isResourceInSearchPath(

strCurrentResourcePath, a\_strResourceSearchPath, a\_bRecursive)) {

filesArray = a\_file.list();

for (int i = 0; i < filesArray.length; i++)

{

// JDK 1.1.8 adds a separator char to the absolute directory path; remove it

String separatorChar = "" + File.separatorChar;

if (a\_file.getAbsolutePath().endsWith(separatorChar))

{

separatorChar = ""; // JDK 1.1.8

}

loadResourcesFromFile(

a\_strResourceSearchPath,

new File(a\_file.getAbsolutePath() + separatorChar + filesArray[i]),

a\_topDirectory, a\_instantiator, a\_loadedResources, a\_bRecursive,

a\_bStopAtFirstResource);

}

}

}

}

/\*\*

\* Gets the absolute URL to a requested resource if the resource is found in the

\* given resource files. In addition to the resource files, the absolute URLs and file types

\* of the resource files must be given. If the correct values are unknown, the must be

\* replaced by "null".

\* @param a\_strRelativeResourcePath the relative path to a resource

\* @param a\_resourceFiles the resource files

\* @param a\_resourceURLs the absolute URLs to the resource files

\* @param a\_resourceTypes the file types of the resource files, either SYSTEM\_RESOURCE\_TYPE\_ZIP

\* or SYSTEM\_RESOURCE\_TYPE\_FILE

\* @return URL

\*/

private static URL getResourceURL(String a\_strRelativeResourcePath,

Vector a\_resourceFiles,

Vector a\_resourceURLs,

Vector a\_resourceTypes)

{

File classPathFile;

String strRelativeResourcePath;

String classPathResourceURL;

Enumeration resourceFiles = a\_resourceFiles.elements();

Class firstClassFound;

FileTypeInstantiator instantiator = (new ResourceLoader()).createFileTypeInstantiator();

Hashtable resourceType;

for (int i = 0; resourceFiles.hasMoreElements(); i++)

{

classPathFile = (File) resourceFiles.nextElement();

classPathResourceURL = (String) a\_resourceURLs.elementAt(i);

if (classPathResourceURL == null)

{

// this resource file has not been searched for classes before

firstClassFound = ClassUtil.getFirstClassFound(classPathFile);

if (firstClassFound == null)

{

// no classes have been found in this resource file;

a\_resourceURLs.setElementAt(RESOURCE\_NO\_CLASSES\_FOUND, i);

continue;

}

strRelativeResourcePath = ClassUtil.toRelativeResourcePath(firstClassFound);

// get the resource type

resourceType = new Hashtable();

loadResources(strRelativeResourcePath, classPathFile, instantiator,

false, true, resourceType);

a\_resourceTypes.setElementAt(resourceType.elements().nextElement(), i);

// extract the URL to the resource directory from the parent class directory

strRelativeResourcePath = "/" + strRelativeResourcePath;

classPathResourceURL =

firstClassFound.getResource(strRelativeResourcePath).toString();

if (!classPathResourceURL.endsWith(strRelativeResourcePath))

{

// this should never happen...

continue;

}

classPathResourceURL = classPathResourceURL.substring(

0, classPathResourceURL.lastIndexOf(strRelativeResourcePath));

a\_resourceURLs.setElementAt(classPathResourceURL, i);

}

else if (classPathResourceURL.trim().equals(RESOURCE\_NO\_CLASSES\_FOUND))

{

// no classes have been found in this resource file

continue;

}

if (a\_resourceTypes.elementAt(i).equals(SYSTEM\_RESOURCE\_TYPE\_FILE))

{

// this resource file is a directory

File testfile = new File(classPathFile, replaceFileSeparatorsSystemSpecific(

a\_strRelativeResourcePath));

// test if the file exists in this directory

if (!testfile.exists())

{

// the requested resource does not exist in this directory

continue;

}

}

else

{

// this resource file is a zip file; test if the resource exists in this file

try

{

if (new ZipFile(classPathFile).getEntry(a\_strRelativeResourcePath)

!= null)

{

/\*\*

\* The requested resource has been found. This contruction might look

\* a bit complicated, but it is needed for JView under Windows.

\* If we test "==null" followed by "continue", the virtual machine stops

\* without any messages.

\*/

}

else

{

// the requested resource does not exist in this zip file

continue;

}

}

catch (Exception a\_e)

{

continue;

}

}

// prepare the relative resource path

if (!a\_strRelativeResourcePath.startsWith("/"))

{

a\_strRelativeResourcePath = "/" + a\_strRelativeResourcePath;

}

// construct the URL for the file

try

{

return new URL(classPathResourceURL + a\_strRelativeResourcePath);

}

catch (MalformedURLException a\_e)

{

// should never happen

}

}

return null;

}

/\*\*

\* Gets the relative resource path for the currently parsed file.

\* @param a\_currentFile the currently parsed file

\* @param a\_topDirectory the top directory in that this file resides; the resource path is

\* given relative to this directory

\* @return the relative resource path

\*/

private static String getCurrentResourcePath(File a\_currentFile, File a\_topDirectory)

{

if( a\_currentFile.toString().equals(a\_topDirectory.toString()))

{

return "/";

}

String strCurrentFile;

int separator = 1;

// JDK 1.1.8 adds a separator after the directory

if (a\_topDirectory.toString().endsWith(File.separator))

{

separator = 0; // JDK 1.1.8

}

strCurrentFile = a\_currentFile.toString().substring(

a\_topDirectory.toString().length() + separator, a\_currentFile.toString().length());

strCurrentFile = strCurrentFile.replace('\\','/');

if (a\_currentFile.isDirectory() && !strCurrentFile.endsWith("/"))

{

strCurrentFile = strCurrentFile + "/";

}

return strCurrentFile;

}

/\*\*

\* Gets the relative resource path for the currently parsed zip entry.

\* @param a\_currentEntry the currently parsed zip entry

\* @return the relative resource path

\*/

private static String getCurrentResourcePath(ZipEntry a\_currentEntry)

{

if (a\_currentEntry.isDirectory() && !a\_currentEntry.toString().endsWith("/"))

{

return a\_currentEntry.toString() + "/";

}

return a\_currentEntry.toString();

}

/\*\*

\* Tests if the currently parsed resource is in the search path.

\* @param a\_strCurrentResourcePath the currently parsed resource

\* @param a\_strResourceSearchPath the resource search path

\* @param a\_bRecursive true if subdirectories should be visited; false otherwise

\* @return true if the current resource path is in the search path; false otherwise

\*/

private static boolean isResourceInSearchPath(String a\_strCurrentResourcePath,

String a\_strResourceSearchPath,

boolean a\_bRecursive)

{

if (a\_strCurrentResourcePath.equals(a\_strResourceSearchPath) ||

a\_strCurrentResourcePath.equals("/"))

{

// the entry was found or this is the directory itself

return true;

}

if (a\_strResourceSearchPath.equals("/"))

{

if (a\_bRecursive)

{

// all resources in the file are loaded

return true;

}

if (a\_strCurrentResourcePath.indexOf("/") >= 0)

{

// the search is not recursive and this is a directory or a file in a subdirectory

return false;

}

}

if (a\_strCurrentResourcePath.length() <= a\_strResourceSearchPath.length())

{

// this cannot be the wanted entry (don't remove, important for zip files)

return false;

}

if (a\_strCurrentResourcePath.startsWith(a\_strResourceSearchPath))

{

// OK, the entry in the search path; now test if it is in a subdirectory

if (a\_strResourceSearchPath.endsWith("/"))

{

if (a\_bRecursive)

{

return true;

}

if (a\_strCurrentResourcePath.substring(

a\_strResourceSearchPath.length()).indexOf("/") < 0)

{

return true;

}

}

}

return false;

}

/\*\*

\* Reformats a given resource name in a way it can be easily interpreted by the resource

\* loader methods. The resulting paths are either null or of the form

\* "dir1/dir2/resource", "dir1/dir2/" or "/". All other cases, including going up directory

\* trees by inserting "../", are handled by this method and transformed to one of the four

\* defined forms.

\* @param a\_strRelativeResourcePath the relative path to a resource

\* @return the formatted file name or null if the file name is illegal

\*/

private static String formatResourcePath(String a\_strRelativeResourcePath)

{

int index, tempIndex;

String temp;

if (a\_strRelativeResourcePath == null)

{

return null;

}

// trim leading and ending white spaces and replace file separators as needed

a\_strRelativeResourcePath = a\_strRelativeResourcePath.trim().replace('\\', '/');

if (a\_strRelativeResourcePath.equals("/"))

{

// this path specifies all resources contained in it

return a\_strRelativeResourcePath;

}

if (a\_strRelativeResourcePath.length() == 0 || a\_strRelativeResourcePath.startsWith("/"))

{

// invalid relative path

return null;

}

// interpret all "/../" as going up the tree

while ((index = a\_strRelativeResourcePath.indexOf("/" + DIR\_UP)) >= 0)

{

if (a\_strRelativeResourcePath.startsWith(DIR\_UP))

{

// invalid relative path

return null;

}

temp = a\_strRelativeResourcePath.substring(0, index);

if ((tempIndex = temp.lastIndexOf("/")) >= 0)

{

temp = temp.substring(0, tempIndex + 1);

}

else

{

temp = "/";

}

temp += a\_strRelativeResourcePath.substring(index + ("/" + DIR\_UP).length(),

a\_strRelativeResourcePath.length());

a\_strRelativeResourcePath = temp;

while (a\_strRelativeResourcePath.startsWith("/"))

{

if (a\_strRelativeResourcePath.equals("/"))

{

break;

}

a\_strRelativeResourcePath = a\_strRelativeResourcePath.substring(

1, a\_strRelativeResourcePath.length());

}

}

if (a\_strRelativeResourcePath.startsWith(DIR\_UP))

{

return null;

}

/\*

while ((index = a\_strRelativeResourcePath.lastIndexOf(DIR\_CURRENT)) >= 0)

{

if (a\_strRelativeResourcePath.equals(DIR\_CURRENT))

{

a\_strRelativeResourcePath = "";

}

else if (index == 0)

{

a\_strRelativeResourcePath =

a\_strRelativeResourcePath.substring(index + DIR\_CURRENT.length(),

a\_strRelativeResourcePath.length());

}

else if (a\_strRelativeResourcePath.charAt(index - 1) == '/')

{

temp = a\_strRelativeResourcePath.substring(0, index) +

a\_strRelativeResourcePath.substring(index + DIR\_CURRENT.length(),

a\_strRelativeResourcePath.length());

a\_strRelativeResourcePath = temp;

}

}\*/

return a\_strRelativeResourcePath;

}

/\*\*

\* Trims a byte array in a way that all bytes after the given length <code> a\_maxLength </code>

\* are cut off.

\* Afterwards, a new byte array is constructed with the bytes from the given

\* <code> a\_arrayToAppendTo </code> and the trimmed array. If <code> a\_maxLength </code> is

\* smaller than or equal to zero a reference to <code> a\_arrayToAppendTo </code> is returned.

\* @param a\_trimmedArray the byte array to trim

\* @param a\_maxLength the maximum length of the trimmed byte array

\* @param a\_arrayToAppendTo the array to append the trimmed byte array to

\* @return the concatenated array

\*/

private static byte[] trimByteArray(byte[] a\_trimmedArray, int a\_maxLength,

byte[] a\_arrayToAppendTo)

{

byte[] temp;

if (a\_maxLength <= 0)

{

temp = a\_arrayToAppendTo;

}

else

{

int trimmedLength;

if (a\_trimmedArray.length > a\_maxLength)

{

trimmedLength = a\_maxLength;

}

else

{

trimmedLength = a\_trimmedArray.length;

}

temp = new byte[a\_arrayToAppendTo.length + trimmedLength];

System.arraycopy(a\_arrayToAppendTo, 0, temp, 0, a\_arrayToAppendTo.length);

System.arraycopy(a\_trimmedArray, 0, temp, a\_arrayToAppendTo.length, trimmedLength);

}

return temp;

}

/\*\*

\* Reads all resources from the classpath and stores them as files.

\* The method does nothing if the classpath has not changed since the last call.

\* @return all resources from the classpath as files

\*/

private static Vector readFilesFromClasspath()

{

String classpath = System.getProperty("java.class.path");

if (ms\_classpath == null || !ms\_classpath.equals(classpath))

{

synchronized (ms\_classpathResourceURLs)

{

StringTokenizer tokenizer;

ms\_classpath = classpath;

ms\_classpathFiles = new Vector();

ms\_classpathResourceURLs = new Vector();

ms\_classpathResourceTypes = new Vector();

tokenizer = new StringTokenizer(ms\_classpath, System.getProperty("path.separator"));

while (tokenizer.hasMoreTokens())

{

ms\_classpathFiles.addElement(

new File(new File(tokenizer.nextToken()).getAbsolutePath()));

ms\_classpathResourceURLs.addElement((Class)null);

ms\_classpathResourceTypes.addElement((String)null);

}

}

}

return ms\_classpathFiles;

}

/\*\*

\* Returns a new ByteArrayInstantiator.

\* This method is needed due to static centext restrictions.

\* @return a new ByteArrayInstantiator

\*/

private ByteArrayInstantiator createByteArrayInstantiator()

{

return new ByteArrayInstantiator();

}

/\*\*

\* Returns a new FileTypeInstantiator.

\* This method is needed due to static centext restrictions.

\* @return a new FileTypeInstantiator

\*/

private FileTypeInstantiator createFileTypeInstantiator()

{

return new FileTypeInstantiator();

}

/\*\*

\* This class is used to get resources as byte arrays.

\*/

private final class ByteArrayInstantiator implements ResourceInstantiator

{

public Object getInstance(File a\_file, File a\_topDirectory)

throws Exception

{

return getStreamAsBytes(new FileInputStream(a\_file));

}

public Object getInstance(ZipEntry a\_entry, ZipFile a\_file)

throws Exception

{

return getStreamAsBytes(a\_file.getInputStream(a\_entry));

}

}

/\*\*

\* Does not load or instantiate resources but returns the file type of resources. The file

\* type may either be SYSTEM\_RESOURCE\_TYPE\_ZIP or SYSTEM\_RESOURCE\_TYPE\_FILE.

\*/

private final class FileTypeInstantiator implements ResourceInstantiator

{

public Object getInstance(File a\_file, File a\_topDirectory)

{

return SYSTEM\_RESOURCE\_TYPE\_FILE;

}

public Object getInstance(ZipEntry a\_entry, ZipFile a\_file)

{

return SYSTEM\_RESOURCE\_TYPE\_ZIP;

}

}

}

**URLDecoder.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

JP iNFOTeCH

\*/

package anon.util;

import java.io.UnsupportedEncodingException;

public class URLDecoder

{

/\*\*

\* Decodes a URL to a unicode. This method uses UTF-8-encoding.

\* @param a\_strURL a URL in UTF-8-encoding

\* @return the URL in unicode encoding

\*/

public static String decode(String a\_strURL)

{

if (a\_strURL == null)

{

return null;

}

StringBuffer output = new StringBuffer();

byte[] enc = new byte[a\_strURL.length()];

int bytes = 0;

int i = 0;

char c;

try

{

while (i < a\_strURL.length())

{

c = a\_strURL.charAt(i);

if (c == '+')

{

output.append(' ');

}

else if (c == '%')

{

enc[bytes] = (byte) Integer.parseInt(a\_strURL.substring(i + 1, i + 3), 16);

bytes++;

i += 2;

}

else

{

output.append(c);

}

i++;

if ( (i < a\_strURL.length() && a\_strURL.charAt(i) != '%') || i >= a\_strURL.length())

{

output.append(new String(enc, 0, bytes, "UTF-8"));

bytes = 0;

}

}

}

catch (NumberFormatException a\_e)

{

return null;

}

catch (UnsupportedEncodingException a\_e)

{

return null;

}

return output.toString();

}

}

**GUI**

**ImageIconLoader.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*

JP iNFOTeCH

\*/

package gui;

import java.util.Hashtable;

import java.awt.Image;

import java.awt.MediaTracker;

import javax.swing.ImageIcon;

import anon.util.ResourceLoader;

//import logging.LogHolder;

//import logging.LogLevel;

//import logging.LogType;

/\*\*

\* This class loads resources from the file system.

\*/

final public class ImageIconLoader

{

// all loaded icons are stored in the cache and do not need to be reloaded from file

private static Hashtable ms\_iconCache = new Hashtable();

private ImageIconLoader()

{

}

/\*\*

\* Loads an ImageIcon from the classpath or the current directory.

\* The icon may be contained in an archive (JAR) or a directory structure. If the icon could

\* not be found in the classpath, it is loaded from the current directory.

\* Once an icon is loaded, it is stored in a memory cache, so that further calls of this method

\* do not load the icon from the file system, but from the cache.

\* @param a\_strRelativeImagePath the relative resource path or filename of the Image

\* @return the loaded ImageIcon or null if the icon could not be loaded

\* (getImageLoadStatus() == java.awt.MediaTracker.ERRORED)

\*/

public static ImageIcon loadImageIcon(String a\_strRelativeImagePath)

{

return loadImageIcon(a\_strRelativeImagePath, true);

}

/\*\*

\* Loads an ImageIcon from the classpath or the current directory.

\* The icon may be contained in an archive (JAR) or a directory structure. If the icon could

\* not be found in the classpath, it is loaded from the current directory.

\* Once an icon is loaded, it is stored in a memory cache, so that further calls of this method

\* do not load the icon from the file system, but from the cache.

\* The image may be loaded synchronously so that the method only returns when the image has been

\* loaded completely (or an error occured), or asynchronously so that the method returns even if

\* the image has not been loaded yet.

\* @param a\_strRelativeImagePath the relative resource path or filename of the Image

\* @param a\_bSync true if the image is loaded synchronously; false otherwise

\* @return the loaded ImageIcon or null if the icon could not be loaded

\* (getImageLoadStatus() == java.awt.MediaTracker.ERRORED)

\*/

public static ImageIcon loadImageIcon(String a\_strRelativeImagePath, boolean a\_bSync)

{

ImageIcon img;

int statusBits;

// try to load the image from the cache

if (ms\_iconCache.containsKey(a\_strRelativeImagePath))

{

return new ImageIcon((Image)ms\_iconCache.get(a\_strRelativeImagePath));

}

// load image from the local classpath or the local directory

try

{

img = new ImageIcon(ResourceLoader.getResourceURL(a\_strRelativeImagePath));

}

catch (NullPointerException a\_e)

{

img = null;

}

if (img != null)

{

if (a\_bSync)

{

statusBits = MediaTracker.ABORTED | MediaTracker.ERRORED | MediaTracker.COMPLETE;

while ((img.getImageLoadStatus() & statusBits) == 0)

{

Thread.yield();

}

}

// write the image to the cache

ms\_iconCache.put(a\_strRelativeImagePath, img.getImage());

}

statusBits = MediaTracker.ABORTED | MediaTracker.ERRORED;

// if (img == null || (img.getImageLoadStatus() & statusBits) != 0)

// {

// LogHolder.log(LogLevel.INFO, LogType.GUI,

// "Could not load requested image '" + a\_strRelativeImagePath + "'!");

// }

return img;

}

}

**JB**

**Dispatcher.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package jb;

import gui.ImageIconLoader;

import java.awt.Image;

import java.io.File;

import java.util.Vector;

import javax.imageio.ImageIO;

import javax.swing.JOptionPane;

/\*\* Dispatcher takes over the control of the VCApplet

\* @author Boßle Johannes

\*

\*/

public class Dispatcher implements Runnable{

/\*\* the Applet which has initiciated this Dispatcher\*/

VCApplet m\_applet;

/\*\* the Thread which is needed while waiting\*/

Thread m\_thread;

/\*\* timout set for waiting for a picture\*/

int m\_timeout = 1000;

/\*\* the image to load\*/

Image m\_loadImage, m\_loadImage2;

/\*\* the Encryptor, which does the encryption\*/

Encryptor m\_encryptor;

/\*\* Constant for the type recognition of the EncryptionType

\* Here it is a 2 out of 2 Scheme

\*/

final static int VCTYPE\_2\_2 = 0;

/\*\* Constant for the type recognition of the EncryptionType

\* Here it is a 3 out of 3 Scheme

\*/

final static int VCTYPE\_3\_3 = 1;

/\*\* Constant for the type recognition of the EncryptionType

\* Here it is a 2 out of n Scheme, with n<=5

\*/

final static int VCTYPE\_2\_N = 2;

/\*\* Constant for the type recognition of the EncryptionType

\* Here it is a 3 out of n Scheme, with n<=5

\*/

final static int VCTYPE\_3\_N = 3;

/\*\*the width of the pic that is used\*/

final int WIDTH = 100;

/\*\*the height of the pic that is used\*/

final int HEIGHT = 100;

/\*\*the strings of the pics which are used if someone not wants to

\* use own pictures

\*/

final String PIC = "smile.gif";

final String GREYPIC ="monalisa.jpg";

final String COLORPIC ="italy.gif";

/\*\* the second picture to load if someone wants to

\* see the same random key effect

\*/

final String PIC2 = "question.gif";

/\*\* which type of vc to use\*/

int m\_vctype;

/\*\* set the maximum Foils\*/

int m\_nrOfFoils;

/\*\* the number of the current shown encrypted pic

\*

\*/

int m\_currentFoil;

/\*\* wasColorOrGrey signalizes, if there has been another example picture

\*

\*/

boolean wasColorOrGrey = false;

/\*\* no option specified\*/

final static int NOOPTION = 0;

/\*\* grey option specified\*/

final static int GREY = 1;

/\*\* colored VC option specified\*/

final static int COLOR = 2;

/\*\* same random key option specified\*/

final static int DOUBLEDKEY = 3;

/\*\* VC with General Access option specified\*/

final static int GENERALACCESS = 4;

/\*\* hold the specified option\*/

int m\_option = NOOPTION;

/\*\* some view elements \*/

ImagePanel m\_srcImagePanel = new ImagePanel(WIDTH,HEIGHT);

ImagePanel m\_srcImagePanel2 = new ImagePanel(WIDTH,HEIGHT);

ImagePanel m\_encImagePanel = new ImagePanel(WIDTH,HEIGHT,this);

ImagePanel m\_encImagePanel2 = new ImagePanel(WIDTH,HEIGHT,this);

ImagePanel m\_ovlImagePanel = new ImagePanel(WIDTH,HEIGHT,this);

/\*\*creates a new Dispatcher-Object\*/

public Dispatcher(VCApplet a){

m\_applet = a;

}

/\*\*loads an given image

\*

\*

\*/

public void loadImage(File f){

// load image

try{

if(f==null){

// change the example pictures for grey and color

if(m\_option==COLOR){

m\_loadImage = ImageIconLoader.loadImageIcon(COLORPIC).getImage();

}else if(m\_option==GREY){

m\_loadImage = ImageIconLoader.loadImageIcon(GREYPIC).getImage();

}else{

m\_loadImage = ImageIconLoader.loadImageIcon(PIC).getImage();

// m\_loadImage = ImageIO.read(new URL(m\_applet.getCodeBase()+PIC)).getSubimage(0,0,WIDTH,HEIGHT);

}

if(m\_loadImage==null){

throw new Exception();

}

m\_loadImage = m\_loadImage.getScaledInstance(WIDTH,HEIGHT,Image.SCALE\_SMOOTH);

} else {

//try to read out the given file

m\_loadImage = ImageIO.read(f).getScaledInstance(WIDTH,HEIGHT,Image.SCALE\_SMOOTH);

// getSubimage(0,0,WIDTH,HEIGHT);

}

}catch (Exception e) {

e.printStackTrace(System.out);

JOptionPane.showMessageDialog(m\_applet,"Loading of the image failed.\n" +

"Please be sure to have specified the right path to the images.\n" +

"The image will be scaled to 100\*100 pixel.","Alert",JOptionPane.ERROR\_MESSAGE);

}

// wait until image loaded

System.out.print("Dispatcher: loading image");

m\_thread = java.lang.Thread.currentThread();

do {

Thread t = new Thread (this);

t.run();

} while (m\_loadImage.getHeight(m\_applet) < 0 & m\_timeout > 0);

if (m\_timeout > 0) {

this.initNewMode();

} else {

System.out.println("Dispatcher: timeout while loading image");

}

}

/\*\* inits a new Mode when a new mode is choosen

\* chooses the right encryptor class

\*

\*/

public void initNewMode(){

System.out.println("Dispatcher: loading image complete");

if(m\_option==GENERALACCESS){

m\_applet.checkBoxPanel.setVisible(true);

}else{

m\_applet.checkBoxPanel.setVisible(false);

}

m\_srcImagePanel.setImage(m\_loadImage,WIDTH,HEIGHT);

m\_applet.inputPanelLeft.add(m\_srcImagePanel2);

// m\_applet.inputPanel.add(srcImagePanel2);

if(m\_option==DOUBLEDKEY){

m\_srcImagePanel2.setVisible(true);

try{

// m\_loadImage2 = ImageIO.read(new URL(m\_applet.getCodeBase()+PIC2)).getSubimage(0,0,WIDTH,HEIGHT);

m\_loadImage2 = ImageIconLoader.loadImageIcon(PIC2).getImage();

}catch(Exception e){

e.printStackTrace(System.out);

}

m\_srcImagePanel2.setImage(m\_loadImage2, WIDTH, HEIGHT);

}

m\_encImagePanel.setImage(null,WIDTH,HEIGHT);

m\_encImagePanel.setVisible(false);

m\_applet.encPanelRight.add(m\_encImagePanel2,0);

m\_encImagePanel2.setImage(null,WIDTH,HEIGHT);

m\_encImagePanel2.setVisible(false);

m\_applet.encPanel.doLayout();

//this.encPanel.doLayout();

m\_ovlImagePanel.setImage(null,WIDTH,HEIGHT);

m\_ovlImagePanel.setVisible(false);

m\_applet.resultPanel.doLayout();

if (m\_loadImage != null) {// encrypt

System.out.println("Dispatcher: choose Encryptor");

m\_encryptor = null;

wasColorOrGrey = false;

switch(m\_vctype){

case VCTYPE\_2\_2 :

if(m\_option==COLOR){

wasColorOrGrey = true;

m\_encryptor = new Enc2\_2\_Color(WIDTH, HEIGHT,2);

}else if(m\_option==DOUBLEDKEY){

m\_encryptor = new EncDoubledKey(m\_loadImage, m\_loadImage2, WIDTH, HEIGHT,2);

// ((EncDoubledKey)m\_encryptor).initEncrypt();

}else if(m\_option==GREY){

wasColorOrGrey = true;

m\_encryptor = new Enc2\_2\_Grey(WIDTH, HEIGHT,2);

}else{

m\_encryptor = new Enc2\_2(WIDTH, HEIGHT,2);

}

break;

case VCTYPE\_3\_3 :

if(m\_option==GENERALACCESS){

m\_encryptor = new EncGA(this.getAccessStructure(),WIDTH, HEIGHT,3);

}else{

m\_encryptor = new Enc3\_3(WIDTH, HEIGHT,3);

}

break;

case VCTYPE\_2\_N : m\_encryptor = new Enc2\_n(WIDTH, HEIGHT,m\_nrOfFoils);

break;

case VCTYPE\_3\_N : m\_encryptor = new Enc3\_n(WIDTH, HEIGHT,m\_nrOfFoils);

break;

}

if(m\_option!=DOUBLEDKEY){

m\_encryptor.initEncrypt(m\_loadImage);

}

}

}

/\*\* sends the encrypted images to the applet

\*

\*/

public void putEncImagesToApplet(){

//send the images to the applet be sure to have choosen the right images

m\_encryptor.encrypt();

// System.out.println("Dispatcher: setting new images");

m\_srcImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerSource()),WIDTH,HEIGHT);

m\_srcImagePanel.setVisible(false);

m\_srcImagePanel.setVisible(true);

m\_srcImagePanel.validate();

if(m\_option==DOUBLEDKEY){

m\_srcImagePanel2.setImage(m\_applet.createImage(((EncDoubledKey)m\_encryptor).getImageProducerSource(true)),WIDTH,HEIGHT);

m\_srcImagePanel2.setVisible(false);

m\_srcImagePanel2.setVisible(true);

m\_srcImagePanel2.validate();

}

if(m\_option==GENERALACCESS){

m\_encryptor = ((EncGA)m\_encryptor).getEncGAWithNewAccessStructure(this.getAccessStructure(),m\_loadImage);

}

if(this.m\_nrOfFoils<3 && m\_option!=DOUBLEDKEY){

m\_applet.prevNextFoilPanel.setVisible(false);

m\_encImagePanel2.setVisible(true);

m\_encImagePanel2.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncrypted(1)),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_encImagePanel2.setVisible(false);

m\_encImagePanel2.setVisible(true);

}else{

m\_applet.prevNextFoilPanel.setVisible(true);

}

m\_encImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncrypted(this.getCurrentFoil())),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_encImagePanel.setVisible(false);

m\_encImagePanel.setVisible(true);

m\_applet.inputPanel.validate();

m\_applet.encPanel.validate();

//try to update resultPanel

this.overlay(m\_applet.getSetForAccess());

m\_applet.resultPanel.setVisible(false);

m\_applet.resultPanel.setVisible(true);

}

/\*\* causes the Encryptor to overlay the foils and sends it to

\* the overlayed pic back to the applet

\*

\*

\*/

public void overlay(boolean[] setForAccess){

if (m\_encryptor != null) {

// System.out.println("Dispatcher: overlay foils");

m\_encryptor.overlayFoils(setForAccess);

m\_ovlImagePanel.setVisible(true);

Image img = m\_applet.createImage(m\_encryptor.getImageProducerOverlay());

m\_ovlImagePanel.setImage(img ,m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_ovlImagePanel.setVisible(false);

m\_ovlImagePanel.setVisible(true);

m\_applet.resultPanel.validate();

m\_applet.resultPanel.setVisible(true);

}

}

/\*\* return the SrcCanvas, the Canvas where the sourceFoil

\* picture can be displayed

\*

\* @return the SourceCanvas

\*/

public ImagePanel getSrcCanvas(){

return m\_srcImagePanel;

}

/\*\* return the encCanvas, the Canvas where the encryptedFoil

\* picture can be displayed

\*

\* @return the encryptedCanvas

\*/

public ImagePanel getEncCanvas(){

return m\_encImagePanel;

}

/\*\*returns the foil with the specified number as an ImagePanel

\*

\* @param nr - the number of the encrypted picture

\* @return the ImagePanel with the encrypted pic of foil number nr

\*/

public ImagePanel getEncCanvas(int nr){

if(m\_option==DOUBLEDKEY){

m\_encImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncrypted(nr)),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else{

m\_encImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncryptedFromVector(nr)),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}

m\_currentFoil = nr;

return m\_encImagePanel;

}

/\*\* return the ResultCanvas, the Canvas where the result

\* picture can be displayed

\*

\* @return the ResultCanvas

\*/

public ImagePanel getResultCanvas(){

return m\_ovlImagePanel;

}

/\*\* some kind of initialization for entering an new VC\_TYPE

\* for the parameters @see Dispatcher

\* @param nrFoils - number of Foils used in this scheme

\* @param option - whether to set an option or not

\* @param type - the number of the scheme

\*/

public void newMode(int nrFoils, int option, int type){

m\_applet.inputFlip.flip(false);

this.reset();

this.setNrOfFoils(nrFoils);

this.setOptions(option);

this.setVctype(type);

m\_applet.enableFunctionsAfterEncrypt(false);

if(m\_option==COLOR || m\_option==GREY || wasColorOrGrey){

this.loadImage(m\_applet.m\_fileImage);

//this.initNewMode is called in loadImage

}else{

this.initNewMode();

}

}

/\*\* sets some values back to their standard value

\*

\*

\*/

public void reset(){

System.out.println("Dispatcher: reset");

if(m\_encImagePanel2.isVisible()){

m\_encImagePanel2.setVisible(false);

}

if(m\_srcImagePanel2.isVisible()){

m\_srcImagePanel2.setVisible(false);

}

this.setCurrentFoil(0);

}

/\*\* wraps the encryptor.getDescription()

\*

\* @return - the description of the encryptor-object

\*/

public String getDescription(){

if(m\_encryptor==null){

System.out.println("encryptor = null");

return new Enc2\_2(WIDTH,HEIGHT,2).getDescription();

}

return m\_encryptor.getDescription();

}

/\*\* Causes a Thread to wait for 100ms while a picture is loaded

\* @see java.lang.Runnable#run()

\*/

public void run() {

try {

Thread.sleep(100);

System.out.print(".");

m\_timeout--;

} catch (InterruptedException e) {

// ready

}

}

/\*\*

\* @return Returns the m\_nrOfFoils.

\*/

public int getNrOfFoils() {

return m\_nrOfFoils;

}

/\*\*

\* @param numberoffoils The numberoffoils to set.

\*/

public void setNrOfFoils(int numberoffoils) {

m\_nrOfFoils = numberoffoils;

}

/\*\*

\* @return Returns the vctype.

\*/

public int getVctype() {

return m\_vctype;

}

/\*\*

\* @param type the vctype to set. for possible values @see{applet#getParameterinfo()}

\*/

public void setVctype(int type) {

m\_vctype = type;

System.out.println("dispatcher: setVCType=" + type);

}

/\*\* sets an option to this dispatcher-Object

\*

\* @param option - for possible values @see Dispatcher

\*/

public void setOptions(int option){

this.m\_option = option;

}

/\*\* sends back the specified AccessStructure for this scheme

\* only wraps the VCApplet#getAccessStructure()

\* @return the boolean[] with the access structure

\*/

public boolean[] getAccessStructure(){

return m\_applet.getAccessStructure();

}

/\*\*

\* @return Returns the m\_currentFoil.

\*/

public int getCurrentFoil() {

return m\_currentFoil;

}

/\*\*

\* @param foil The m\_currentFoil to set.

\*/

public void setCurrentFoil(int foil) {

m\_currentFoil = foil;

}

/\*\* loads the encrypted foils

\*

\* @param files - the encrypted foils to load

\* @return - true if successful, false otherwise

\*/

public boolean loadAsEncFoil(File[] files){

boolean success = false;

for(int i=0; i<files.length; i++){

if(files[i]==null)return false;

}

try{

int width = m\_encryptor.getWidthEnc();

int height = m\_encryptor.getHeightEnc();

Vector v = m\_encryptor.getM\_Foils();

v.clear();

for(int i=0; i<files.length; i++){

Image image = ImageIO.read(files[i]).getSubimage(0,0,width,height);

int[] tempPix = new int[width \* height];

m\_encryptor.grabImage(image,tempPix,width,height);

Foil f = new Foil(tempPix,width,height);

v.add(i,f);

}

//set the images to the Applet

if(this.m\_nrOfFoils<3){

m\_applet.prevNextFoilPanel.setVisible(false);

m\_encImagePanel2.setVisible(true);

Image im = m\_applet.createImage(m\_encryptor.getImageProducerEncryptedFromVector(1));

m\_encImagePanel2.setImage(im,m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else{

m\_applet.prevNextFoilPanel.setVisible(true);

}

Image img = m\_applet.createImage(m\_encryptor.getImageProducerEncryptedFromVector(0));

m\_encImagePanel.setImage(img,m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_applet.inputPanel.validate();

m\_applet.encFlip.flip(true);

m\_applet.encFlip.flip(false);

m\_applet.encPanel.validate();

success = true;

}catch(Exception e){

e.printStackTrace(System.out);

}

return success;

}

/\*\* method to save the current foil as a picture

\*

\*

\*/

public void saveCurrent(){

if(m\_ovlImagePanel.isMarked()){

m\_ovlImagePanel.save();

return;

}

if(this.getNrOfFoils()==2 && m\_option!=DOUBLEDKEY){

//which foil is marked

if(m\_encImagePanel.isMarked() && !m\_encImagePanel2.isMarked()){

m\_encImagePanel.save();

}else if(!m\_encImagePanel.isMarked() && m\_encImagePanel2.isMarked()){

m\_encImagePanel2.save();

}else{

JOptionPane.showMessageDialog(m\_applet,"Please select one transparency by left click with the mouse" +

"\nor do a right click on the transparency to save directly.","Warning",JOptionPane.WARNING\_MESSAGE);

}

}else{

m\_encImagePanel.save();

}

m\_applet.markCurrentShowButton(m\_applet.SELECTEDCOLOR);

}

/\*\* method to save all foils as pictures

\*

\*

\*/

public void saveAll(){

int nr = this.getNrOfFoils();

if(m\_option == Dispatcher.DOUBLEDKEY){

nr++;

for(int i=0; i<nr; i++){

m\_encImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncrypted(i)),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_encImagePanel.save();

}

m\_encImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncrypted(this.getCurrentFoil())),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_applet.markCurrentShowButton(m\_applet.SELECTEDCOLOR);

return;

}

for(int i=0; i<nr; i++){

m\_encImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncryptedFromVector(i)),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_encImagePanel.save();

}

m\_encImagePanel.setImage(m\_applet.createImage(m\_encryptor.getImageProducerEncryptedFromVector(this.getCurrentFoil())),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

m\_applet.markCurrentShowButton(m\_applet.SELECTEDCOLOR);

}

/\*\* call a new Printer-Object and let it do the necessary

\* work to get a printed paper.

\*

\*/

public void print(){

if(m\_ovlImagePanel.isMarked()){

new Printer(0,0,m\_encryptor.getImageProducerOverlay(),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

return;

}

if(this.getNrOfFoils()==2 && m\_option!=DOUBLEDKEY){

if(m\_encImagePanel.isMarked() && !m\_encImagePanel2.isMarked()){

new Printer(1,2,m\_encryptor.getImageProducerEncryptedFromVector(0),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else if(!m\_encImagePanel.isMarked() && m\_encImagePanel2.isMarked()){

new Printer(2,2,m\_encryptor.getImageProducerEncryptedFromVector(1),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else{

JOptionPane.showMessageDialog(m\_applet,"Please select one transparency by left click with the mouse.","Warning",JOptionPane.WARNING\_MESSAGE);

}

}else{

if(m\_option != DOUBLEDKEY){

new Printer((this.getCurrentFoil()+1),this.getNrOfFoils(),m\_encryptor.getImageProducerEncryptedFromVector(this.getCurrentFoil()),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else{

new Printer((this.getCurrentFoil()+1),this.getNrOfFoils(),m\_encryptor.getImageProducerEncrypted(this.getCurrentFoil()),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}

}

m\_applet.markCurrentShowButton(m\_applet.SELECTEDCOLOR);

}

/\*\* creates a new JFrame and shows the selected Transparency

\* in a bigger way

\*

\*/

public void zoom(){

if(m\_ovlImagePanel.isMarked()){

new Zoom("resulting",m\_encryptor.getImageProducerOverlay(),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

return;

}

if(this.getNrOfFoils()==2 && m\_option!=DOUBLEDKEY){

if(m\_encImagePanel.isMarked() && !m\_encImagePanel2.isMarked()){

new Zoom("#1",m\_encryptor.getImageProducerEncryptedFromVector(0),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else if(!m\_encImagePanel.isMarked() && m\_encImagePanel2.isMarked()){

new Zoom("#2",m\_encryptor.getImageProducerEncryptedFromVector(1),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else{

JOptionPane.showMessageDialog(m\_applet,"Please select one transparency by left click with the mouse.","Warning",JOptionPane.WARNING\_MESSAGE);

}

}else{

int nr = this.getCurrentFoil();

if(m\_option != DOUBLEDKEY){

new Zoom("#"+(nr+1), m\_encryptor.getImageProducerEncryptedFromVector(nr),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}else{

new Zoom("#"+(nr+1), m\_encryptor.getImageProducerEncrypted(nr),m\_encryptor.getWidthEnc(),m\_encryptor.getHeightEnc());

}

}

m\_applet.markCurrentShowButton(m\_applet.SELECTEDCOLOR);

}

/\*\* deselects a marked ImagePanel if another one is marked

\*

\* @param ip - the imagePanel with the newer mark

\*/

public void setOtherImagePanelUnmarked(ImagePanel ip){

boolean enc = m\_encImagePanel.isMarked();

boolean enc1 = m\_encImagePanel2.isMarked();

boolean ovl = m\_ovlImagePanel.isMarked();

if((this.getNrOfFoils()==2) &&((enc&&enc1)||(enc&&ovl)||(enc1&&ovl))){

//if two transparancies and two of them are marked

if(m\_encImagePanel==ip){

//the newer mark is in encImagePanel

m\_encImagePanel2.setMarked(false);

m\_ovlImagePanel.setMarked(false);

}else if(m\_encImagePanel2==ip){

m\_encImagePanel.setMarked(false);

m\_ovlImagePanel.setMarked(false);

}else{

m\_encImagePanel.setMarked(false);

m\_encImagePanel2.setMarked(false);

}

}else if((enc&&ovl)){

if(m\_encImagePanel==ip){

m\_ovlImagePanel.setMarked(false);

}else{

m\_encImagePanel.setMarked(false);

}

}

}

}

**Enc2\_2.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package jb;

public class Enc2\_2 extends Encryptor {

/\*\* description of this VCMode\*/

final String DESCRIPTION = "This is the basic mode for the visual" +

" cryptography introduced by Naor and Shamir in 1994. This " +

"cryptographic scheme is perfectly secure and can be " +

"encoded without any computations directly by the human " +

"visual system. It conceals a picture by generating two " +

"transparencies which look like random noise. But " +

"overlaying of these transparencies leads to a \"clear\" " +

"image, which contains the original information. For more " +

"possible schemes with more features look at the mode menu." +

"<br><b>How is it done?</b>" +

"<br>Each pixel is divided into 4 subpixel, which can be black or white. " +

"There are always 2 black and 2 white subpixel. <br>" +

"For displaying a white pixel, the sequence of the subpixel " +

"on the first and on the second transparency are identically. " +

"Overlaying leads to 2 black and 2 white subpixel.<br>" +

"For displaying a black subpixel the sequences of the subpixel " +

"are different, so that you get 4 black subpixel by overlaying.";

/\*\* creates a new encryptor object for the VC with 2 out of 2<br/>

\* mainly it sets up the init-matrices for this scheme

\*

\* @param height - height of the source pic

\* @param width - width of the source pic

\* @param n - number of foils

\*/

public Enc2\_2(int height, int width, int n){

super(height, width, 2);

// init matrices

m\_initMatrixC0 = new IntMatrix(m\_maxFoil, m\_maxSubpixel);

m\_initMatrixC1 = new IntMatrix(m\_maxFoil, m\_maxSubpixel);

m\_initMatrixC0.setElement(0, 0, WHITEPIXEL);

m\_initMatrixC0.setElement(0, 1, WHITEPIXEL);

m\_initMatrixC0.setElement(0, 2, BLACKPIXEL);

m\_initMatrixC0.setElement(0, 3, BLACKPIXEL);

m\_initMatrixC0.setElement(1, 0, WHITEPIXEL);

m\_initMatrixC0.setElement(1, 1, WHITEPIXEL);

m\_initMatrixC0.setElement(1, 2, BLACKPIXEL);

m\_initMatrixC0.setElement(1, 3, BLACKPIXEL);

m\_initMatrixC1.setElement(0, 0, WHITEPIXEL);

m\_initMatrixC1.setElement(0, 1, WHITEPIXEL);

m\_initMatrixC1.setElement(0, 2, BLACKPIXEL);

m\_initMatrixC1.setElement(0, 3, BLACKPIXEL);

m\_initMatrixC1.setElement(1, 0, BLACKPIXEL);

m\_initMatrixC1.setElement(1, 1, BLACKPIXEL);

m\_initMatrixC1.setElement(1, 2, WHITEPIXEL);

m\_initMatrixC1.setElement(1, 3, WHITEPIXEL);

}

/\*\* extracts the pixels for one foil

\* @param foilNr the number of foil which shall be extracted

\* @return Foil with the pixels for the foil

\*/

public Foil getFoil(int foilNr){

int index;

int tempPix[] = new int[m\_hEnc \* m\_wEnc]; // array for grabbing pic

// copy encrypted pic to array

for (int y = 0; y < m\_hSrc; y++) {

for (int x = 0; x < m\_wSrc; x++) {

index = 2 \* (x + y \* m\_hEnc);

tempPix[index] = m\_pixels[x][y].getSubpixel(foilNr, 0);

tempPix[index + 1] = m\_pixels[x][y].getSubpixel(foilNr, 1);

tempPix[index + m\_hEnc] = m\_pixels[x][y].getSubpixel(foilNr, 2);

tempPix[index + m\_hEnc + 1] = m\_pixels[x][y].getSubpixel(foilNr, 3);

}

}

//System.out.println("foilNr: "+foilNr);

return new Foil(tempPix,m\_wEnc, m\_hEnc);

}

// /\*\* extracts the resulting pic for the resultFoil

// \* @return the foil/pic getting by overlaying some encrypted

// \* foils

// \*/

// public Foil getOverlayedPic(){

// int tempPix[] = new int[m\_hEnc \* m\_wEnc]; // array for grabbing pic

// int col, index;

// // copy encrypted pic to array

// for (int y = 0; y < m\_hSrc; y++) {

// for (int x = 0; x < m\_wSrc; x++) {

// index = 2 \* (x + y \* m\_hEnc);

//

// tempPix[index] = m\_pixels[x][y].getOverlaySubpixel(0);

// tempPix[index + 1] = m\_pixels[x][y].getOverlaySubpixel(1);

// tempPix[index + m\_hEnc] = m\_pixels[x][y].getOverlaySubpixel(2);

// tempPix[index + m\_hEnc + 1] = m\_pixels[x][y].getOverlaySubpixel(3);

// }

// }

//

// return new Foil(tempPix, m\_wEnc, m\_hEnc);

// }

/\*\*

\* @see jb.Encryptor#getMaxSubpixel()

\*/

public int getMaxSubpixel() {

return 4;

}

/\*\*

\* @see jb.Encryptor#getPermutationInstance()

\*/

public Permutation getPermutationInstance() {

return new Permutation(m\_maxSubpixel);

}

/\*\*

\* @see jb.Encryptor#getFactorWidth()

\*/

public int getFactorWidth() {

return 2;

}

/\*\*

\* @see jb.Encryptor#getFactorHeight()

\*/

public int getFactorHeight() {

return 2;

}

/\*\*

\* @see jb.Encryptor#getDescription()

\*/

public String getDescription() {

return DESCRIPTION;

}

}

**Enc2\_2\_Color.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package jb;

import java.util.Vector;

/\*\* Encryptor - Object for the colored VisualCryptography<br/>

\* The implemented scheme was introduced by Yang and Laih 2000<br/>

\* It works by defining some colors and deleting some colors through the

\* overlaying process, whenever a black pixel is shown.<br/>

\* for example: red and black = black; red and red = red;

\* @author Boßle Johannes

\*

\*/

public class Enc2\_2\_Color extends Encryptor {

/\*\* color definition as int for a red pixel\*/

final static int REDPIXEL = (255 << 24) | (255 << 16);

/\*\* color definition as int for a green pixel\*/

final static int GREENPIXEL = (255 << 24) |(255 << 8);

/\*\* color definition as int for a blue pixel\*/

final static int BLUEPIXEL = (255 << 24) | 255;

// /\*\* color definition as int for a white pixel\*/

// final static int WHITEPIXEL = (255 << 24) | (255 << 16) | (255 << 8) | 255;

// /\*\* color definition as int for a black pixel\*/

// final static int BLACKPIXEL = (255 << 24);

/\*\* description of this VCMode\*/

final String DESCRIPTION = "creates two transparencies for the visual " +

"cryptography with colors. The original picture is " +

"taken and for each pixel is decided which elementary " +

"color (red, green, blue or black) it has. Each pixel " +

"contains 1 red, 1 green, 1 blue and 3 black subpixel. " +

"By overlaying the transparencies the black color dominates " +

"all others and is shown. For displaying a concrete color " +

"(for example red) all subpixel except the red one are " +

"overlayed with an black subpixel. The red pixel is overlayed " +

"with a red subpixel. So you achieve 5 black " +

"and a red subpixel, and for the human visual system " +

"it appears as a (dark)red pixel.";

/\*\* the threshold of this scheme

\* used to decide the pixel color\*/

final int THRESHOLD = 64;

/\*\* Color-Matrices\*/

IntMatrix m\_initMatrixRed,m\_initMatrixGreen,m\_initMatrixBlue,m\_initMatrixWhite,m\_initMatrixBlack;

/\*\* Array of colored Matrices\*/

IntMatrix[] m\_Red, m\_Green, m\_Blue, m\_White, m\_Black;

/\*\* creates a new encryptor object for the colored VC<br/>

\* mainly it sets up the init-matrices for this scheme

\*

\* @param height - height of the source pic

\* @param width - width of the source pic

\* @param n - number of foils

\*/

public Enc2\_2\_Color(int height, int width, int n){

super(height, width, 2);

m\_initMatrixRed = new IntMatrix(m\_maxFoil, m\_maxSubpixel);

m\_initMatrixGreen = new IntMatrix(m\_maxFoil, m\_maxSubpixel);

m\_initMatrixBlue = new IntMatrix(m\_maxFoil, m\_maxSubpixel);

m\_initMatrixWhite = new IntMatrix(m\_maxFoil, m\_maxSubpixel);

m\_initMatrixBlack = new IntMatrix(m\_maxFoil, m\_maxSubpixel);

//init red

m\_initMatrixRed.setElement(0,0,REDPIXEL);

m\_initMatrixRed.setElement(0,1,GREENPIXEL);

m\_initMatrixRed.setElement(0,2,BLUEPIXEL);

m\_initMatrixRed.setElement(0,3,BLACKPIXEL);

m\_initMatrixRed.setElement(0,4,BLACKPIXEL);

m\_initMatrixRed.setElement(0,5,BLACKPIXEL);

m\_initMatrixRed.setElement(1,0,REDPIXEL);

m\_initMatrixRed.setElement(1,1,BLACKPIXEL);

m\_initMatrixRed.setElement(1,2,BLACKPIXEL);

m\_initMatrixRed.setElement(1,3,BLACKPIXEL);

m\_initMatrixRed.setElement(1,4,GREENPIXEL);

m\_initMatrixRed.setElement(1,5,BLUEPIXEL);

//init green

m\_initMatrixGreen.setElement(0,0,REDPIXEL);

m\_initMatrixGreen.setElement(0,1,GREENPIXEL);

m\_initMatrixGreen.setElement(0,2,BLUEPIXEL);

m\_initMatrixGreen.setElement(0,3,BLACKPIXEL);

m\_initMatrixGreen.setElement(0,4,BLACKPIXEL);

m\_initMatrixGreen.setElement(0,5,BLACKPIXEL);

m\_initMatrixGreen.setElement(1,0,BLACKPIXEL);

m\_initMatrixGreen.setElement(1,1,GREENPIXEL);

m\_initMatrixGreen.setElement(1,2,BLACKPIXEL);

m\_initMatrixGreen.setElement(1,3,BLACKPIXEL);

m\_initMatrixGreen.setElement(1,4,REDPIXEL);

m\_initMatrixGreen.setElement(1,5,BLUEPIXEL);

//init blue

m\_initMatrixBlue.setElement(0,0,REDPIXEL);

m\_initMatrixBlue.setElement(0,1,GREENPIXEL);

m\_initMatrixBlue.setElement(0,2,BLUEPIXEL);

m\_initMatrixBlue.setElement(0,3,BLACKPIXEL);

m\_initMatrixBlue.setElement(0,4,BLACKPIXEL);

m\_initMatrixBlue.setElement(0,5,BLACKPIXEL);

m\_initMatrixBlue.setElement(1,0,BLACKPIXEL);

m\_initMatrixBlue.setElement(1,1,BLACKPIXEL);

m\_initMatrixBlue.setElement(1,2,BLUEPIXEL);

m\_initMatrixBlue.setElement(1,3,BLACKPIXEL);

m\_initMatrixBlue.setElement(1,4,REDPIXEL);

m\_initMatrixBlue.setElement(1,5,GREENPIXEL);

//init white

m\_initMatrixWhite.setElement(0,0,REDPIXEL);

m\_initMatrixWhite.setElement(0,1,GREENPIXEL);

m\_initMatrixWhite.setElement(0,2,BLUEPIXEL);

m\_initMatrixWhite.setElement(0,3,BLACKPIXEL);

m\_initMatrixWhite.setElement(0,4,BLACKPIXEL);

m\_initMatrixWhite.setElement(0,5,BLACKPIXEL);

m\_initMatrixWhite.setElement(1,0,BLACKPIXEL);

m\_initMatrixWhite.setElement(1,1,BLACKPIXEL);

m\_initMatrixWhite.setElement(1,2,BLACKPIXEL);

m\_initMatrixWhite.setElement(1,3,REDPIXEL);

m\_initMatrixWhite.setElement(1,4,GREENPIXEL);

m\_initMatrixWhite.setElement(1,5,BLUEPIXEL);

//init black

m\_initMatrixBlack.setElement(0,0,REDPIXEL);

m\_initMatrixBlack.setElement(0,1,GREENPIXEL);

m\_initMatrixBlack.setElement(0,2,BLUEPIXEL);

m\_initMatrixBlack.setElement(0,3,BLACKPIXEL);

m\_initMatrixBlack.setElement(0,4,BLACKPIXEL);

m\_initMatrixBlack.setElement(0,5,BLACKPIXEL);

m\_initMatrixBlack.setElement(1,0,REDPIXEL);

m\_initMatrixBlack.setElement(1,1,GREENPIXEL);

m\_initMatrixBlack.setElement(1,2,BLUEPIXEL);

m\_initMatrixBlack.setElement(1,3,BLACKPIXEL);

m\_initMatrixBlack.setElement(1,4,BLACKPIXEL);

m\_initMatrixBlack.setElement(1,5,BLACKPIXEL);

}

/\*\* sets up the pixels by giving a matrix and the original colour

\*

\* @param tempPix

\*/

public void setMatrixToPixel(int[] tempPix){

// store grabbed image for encryption

System.out.println(" encryptor special color: setting matrix to each Pixel");

for (int y = 0; y < m\_hSrc; y++) {

for (int x = 0; x < m\_wSrc; x++) {

int index = tempPix[x + y \* m\_hSrc];

m\_pixels[x][y].setColor(index);

int decision = this.decideColor(index);

switch(decision){

/\*case WHITEPIXEL:

m\_pixels[x][y].setMatrix(m\_White[this.getRandom()]);

break;\*/

case REDPIXEL:

m\_pixels[x][y].setMatrix(m\_Red[this.getRandom()]);

break;

case GREENPIXEL:

m\_pixels[x][y].setMatrix(m\_Green[this.getRandom()]);

break;

case BLUEPIXEL:

m\_pixels[x][y].setMatrix(m\_Blue[this.getRandom()]);

break;

/\*case BLACKPIXEL:

m\_pixels[x][y].setMatrix(m\_Black[this.getRandom()]);

break;\*/

}

}

}//end for - store

}//end prepareMatrix

/\*\* decide which elementary color this pixel has

\*

\* @param pixel - the original pixel color

\* @return - the elementary color which can be encoded

\*/

public int decideColor(int pixel){

int result=0;

int red = (pixel >> 16) & 0xff;

int green = (pixel >> 8) & 0xff;

int blue = (pixel ) & 0xff;

if((red<THRESHOLD)&&(green<THRESHOLD)&&(blue<THRESHOLD)){

result = WHITEPIXEL;

}else if((red>3\*THRESHOLD) && (green>3\*THRESHOLD) && (blue>3\*THRESHOLD)){

result = BLACKPIXEL;

}else if((red > 2\*THRESHOLD) && (red> green) && (red>blue)){

result = REDPIXEL;

}else if((green > 2\*THRESHOLD) && (green>red) && (green>blue)){

result = GREENPIXEL;

}else if((blue > 2\*THRESHOLD) && (blue>green) && (blue>red)){

result = BLUEPIXEL;

}else{

result = WHITEPIXEL;

}

return result;

}

/\*\* method to create all possible permutations of a

\* init matrix

\*

\*/

public void doPermutation(){

m\_White = new IntMatrix[m\_maxPerm];

m\_Black = new IntMatrix[m\_maxPerm];

m\_Red = new IntMatrix[m\_maxPerm];

m\_Green = new IntMatrix[m\_maxPerm];

m\_Blue = new IntMatrix[m\_maxPerm];

int[][] orderArray = m\_permutation.getPermArray();

Vector c0 = m\_initMatrixWhite.getPermMatrixVector(orderArray);

Vector c1 = m\_initMatrixBlack.getPermMatrixVector(orderArray);

Vector c2 = m\_initMatrixRed.getPermMatrixVector(orderArray);

Vector c3 = m\_initMatrixGreen.getPermMatrixVector(orderArray);

Vector c4 = m\_initMatrixBlue.getPermMatrixVector(orderArray);

for(int i=0; i<m\_maxPerm; i++){

m\_White[i] = (IntMatrix)c0.get(i);

m\_Black[i] = (IntMatrix)c1.get(i);

m\_Red[i] = (IntMatrix)c2.get(i);

m\_Green[i] = (IntMatrix)c3.get(i);

m\_Blue[i] = (IntMatrix)c4.get(i);

}

}

/\*\*

\* @see jb.Encryptor#getFoil(int)

\*/

public Foil getFoil(int foilNr) {

int index;

int factor = this.getFactorWidth();

int tempPix[] = new int[m\_hEnc \* m\_wEnc]; // array for grabbing pic

// copy encrypted pic to array

for (int y = 0; y < m\_hSrc; y++) {

for (int x = 0; x < m\_wSrc; x++) {

index = factor \* (x + y \* m\_hEnc);

tempPix[index] = m\_pixels[x][y].getSubpixel(foilNr, 0);

tempPix[index + 1] = m\_pixels[x][y].getSubpixel(foilNr, 1);

tempPix[index + 2] = m\_pixels[x][y].getSubpixel(foilNr, 2);

tempPix[index + m\_wEnc] = m\_pixels[x][y].getSubpixel(foilNr, 3);

tempPix[index + m\_wEnc + 1] = m\_pixels[x][y].getSubpixel(foilNr, 4);

tempPix[index + m\_wEnc + 2] = m\_pixels[x][y].getSubpixel(foilNr, 5);

}

}

return new Foil(tempPix, m\_wEnc, m\_hEnc);

}

// /\*\*

// \* @see jb.Encryptor#getOverlayedPic()

// \*/

// public Foil getOverlayedPic() {

// int tempPix[] = new int[m\_hEnc \* m\_wEnc]; // array for grabbing pic

// int col, index;

// int factor = this.getFactorWidth();

// // copy encrypted pic to array

// for (int y = 0; y < m\_hSrc; y++) {

// for (int x = 0; x < m\_wSrc; x++) {

// index = factor \* (x + y \* m\_hEnc);

//

// tempPix[index] = m\_pixels[x][y].getOverlaySubpixel(0);

// tempPix[index + 1] = m\_pixels[x][y].getOverlaySubpixel(1);

// tempPix[index + 2] = m\_pixels[x][y].getOverlaySubpixel(2);

// tempPix[index + m\_wEnc] = m\_pixels[x][y].getOverlaySubpixel(3);

// tempPix[index + m\_wEnc+1] = m\_pixels[x][y].getOverlaySubpixel(4);

// tempPix[index + m\_wEnc+2] = m\_pixels[x][y].getOverlaySubpixel(5);

// }

// }

// return new Foil(tempPix, m\_wEnc, m\_hEnc);

// }

/\*\*

\* @see jb.Encryptor#getMaxSubpixel()

\*/

public int getMaxSubpixel() {

return 6;

}

/\*\*

\* @see jb.Encryptor#getPermutationInstance()

\*/

public Permutation getPermutationInstance() {

return new Permutation(m\_maxSubpixel);

}

/\*\*

\* @see jb.Encryptor#getFactorWidth()

\*/

public int getFactorWidth() {

return 3;

}

/\*\*

\* @see jb.Encryptor#getFactorHeight()

\*/

public int getFactorHeight() {

return 2;

}

/\*\*

\* @see jb.Encryptor#getDescription()

\*/

public String getDescription() {

return DESCRIPTION;

}

}

**VCApplication.java**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package jb;

import javax.swing.JApplet;

import javax.swing.JFrame;

/\*\* run the VCApplet as an application

\*

\*/

public class VCApplication{

static final JApplet applet = new VCApplet();

/\*\*starts the applet in a frame

\*

\* @param argv - the normal argv[] - here it is empty

\*/

static public void main (String argv[]) {

// if(argv.length==1){

// JFrame frame = new JFrame ("Embedded Extended Visual Cryptography Schemes");

// frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// frame.getContentPane().add ("Center", applet);

//// applet.setStub(new VCAppletStub (applet, argv));

// frame.setSize(700,700);

// frame.setVisible(true);

// applet.init();

// applet.start();

//// frame.pack();

// }else{

// //print out the usage term

// System.out.println("usage: java VCApplication host=<path>");

// System.out.println("<path> = path to the pictures to load");

// System.out.println("for windows for example:\n host=\"C:\\Dokumente und Einstellungen\\Boßle Johannes\\Eigene Dateien\\sec\\vc\_work\\vc\"");

// System.out.println("or \"\" if nothing was changed (vc.jar and the pics are in the same directory)");

// System.out.println("trying to continue anyway...");

// String[] arg = {"host=\"\""};

// VCApplication.main(arg);

// }

JFrame frame = new JFrame ("Embedded Extended Visual Cryptography Schemes");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().add ("Center", applet);

// applet.setStub(new VCAppletStub (applet, argv));

frame.setSize(700,740);

frame.setVisible(true);

applet.init();

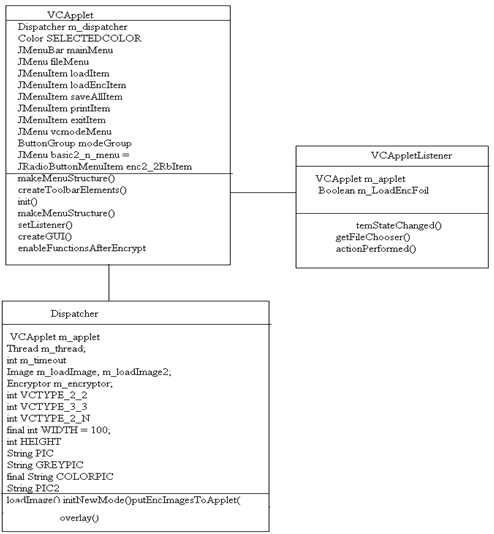
applet.start();

}

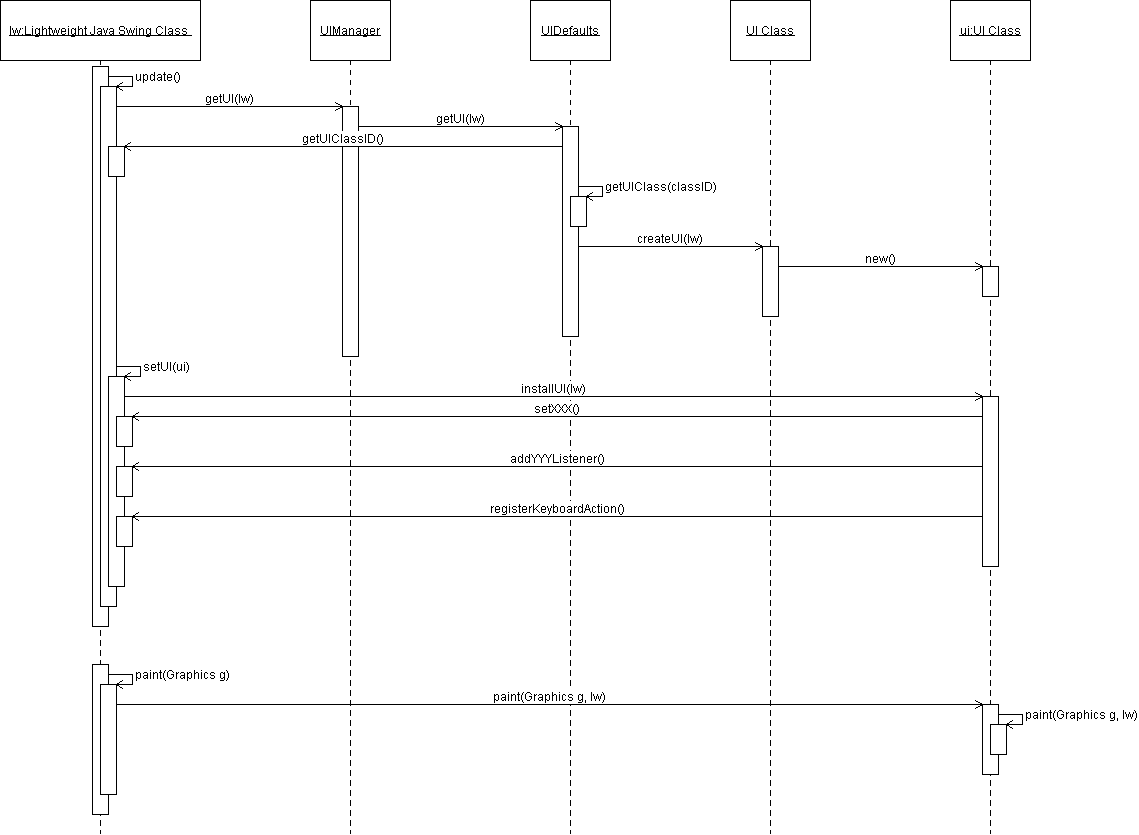
}

**CHAPTER 5: USER INTERACTION DIAGRAM MODULE WISE**

CLASS DIAGRAM



**Sequence Diagram**



**CHAPTER 6: TESTING**

## 

## 6.1 Internal Testing

  **Internal testing** deals with low-level implementation. Here each function or component is tested. This testing is accomplished by the implementation teams. This focus is also called clear-box testing, or sometimes white-box testing, because all details are visible to the test. Internal limits are tested here.

## 6.2 Unit Testing

**Unit testing** deals with testing a unit   as a whole. This would test the interaction of many functions but confine the test within one unit. The exact scope of a unit is left to interpretation. Supporting test code, sometimes called **scaffolding**, may be necessary to support an individual test. This type of testing is driven by the architecture and implementation teams. This focus is also called black-box testing because only the details of the interface are visible to the test. Limits that are global to a unit are tested here.

In the construction industry, scaffolding is a temporary, easy to assemble and disassemble, frame placed around a building to facilitate the construction of the building. The construction workers first build the scaffolding and then the building. Later the scaffolding is removed, exposing the completed building. Similarly, in software testing, one particular test may need some supporting software. This software establishes an environment around the test. Only when this environment is established can a correct evaluation of the test take place. The scaffolding software may establish state and values for data structures as well as providing dummy external functions for the test. Different scaffolding software may be needed from one test to another test. Scaffolding software rarely is considered part of the system.

Sometimes the scaffolding software becomes larger than the system software being tested. Usually the scaffolding software is not of the same quality as the system software and frequently is quite fragile. A small change in the test may lead to much larger changes in the scaffolding.

Internal and unit testing can be automated with the help of coverage tools. A coverage tool analyzes the source code and generates a test that will execute every alternative thread of execution. It is still up to the programmer to combine these test into meaningful cases to validate the result of each thread of execution. Typically, the coverage tool is used in a slightly different way. First the coverage tool is used to augment the source by placing informational prints after each line of code. Then the testing suite is executed generating an audit trail. This audit trail is analyzed and reports the percent of the total system code executed during the test suite. If the coverage is high and the untested source lines are of low impact to the system's overall quality, then no more additional tests are required.

## 6.3 Application Testing

**Application testing** deals with tests for the entire application.   This is driven by the scenarios from the analysis team. Application limits and features are tested here.

The application must successfully execute all scenarios before it is ready for general customer availability. After all, the scenarios are a part of the requirement document and measure success. Application testing represents the bulk of the testing done by industry.

Unlike the internal and unit testing, which are programmed, these test are usually driven by scripts that run the system with a collection of parameters and collect results. In the past, these scripts may have been written by hand but in many modern systems this process can be automated.

Most current applications have graphical user interfaces (GUI). Testing a GUI to assure quality becomes a bit of a problem. Most, if not all, GUI systems have event loops. The GUI event loop contains signals for mouse, keyboard, window, and other related events. Associated with each event are the coordinates on the screen of the event. The screen coordinates can be related back to the GUI object and then the event can be serviced. Unfortunately, if some GUI object is positioned at a different location on the screen, then the coordinates change in the event loop. Logically the events at the new coordinates should be associated with the same GUI object. This logical association can be accomplished by giving unique names to all of the GUI objects and providing the unique names as additional information in the events in the event loop. The GUI application reads the next event off of the event loop, locates the GUI object, and services the event.

The events on the event loop are usually generated by human actions such as typing characters, clicking mouse buttons, and moving the cursor. A simple modification to the event loop can journal the events into a file. At a later time, this file could be used to regenerate the events, as if the human was present, and place them on the event loop. The GUI application will respond accordingly. A tester, using the GUI, now executes a scenario. A journal of the GUI event loop from the scenario is captured. At a later time the scenario can be repeated again and again in an automated fashion. The ability to repeat a test is key to automation and stress testing.

## 6.4 Stress Testing

**Stress testing** deals with the quality of the application in the environment.   The idea is to create an environment more demanding of the application than the application would experience under normal work loads. This is the hardest and most complex category of testing to accomplish and it requires a joint effort from all teams.

A test environment is established with many testing stations. At each station, a script is exercising the system. These scripts are usually based on the regression suite. More and more stations are added, all simultaneous hammering on the system, until the system breaks. The system is repaired and the stress test is repeated until a level of stress is reached that is higher than expected to be present at a customer site.

Race conditions and memory leaks are often found under stress testing. A race condition is a conflict between at least two tests. Each test works correctly when done in isolation. When the two tests are run in parallel, one or both of the tests fail. This is usually due to an incorrectly managed lock.

A memory leak happens when a test leaves allocated memory behind and does not correctly return the memory to the memory allocation scheme. The test seems to run correctly, but after being exercised several times, available memory is reduced until the system fails.

**CHAPTER 7: SCREEN SHOTS**

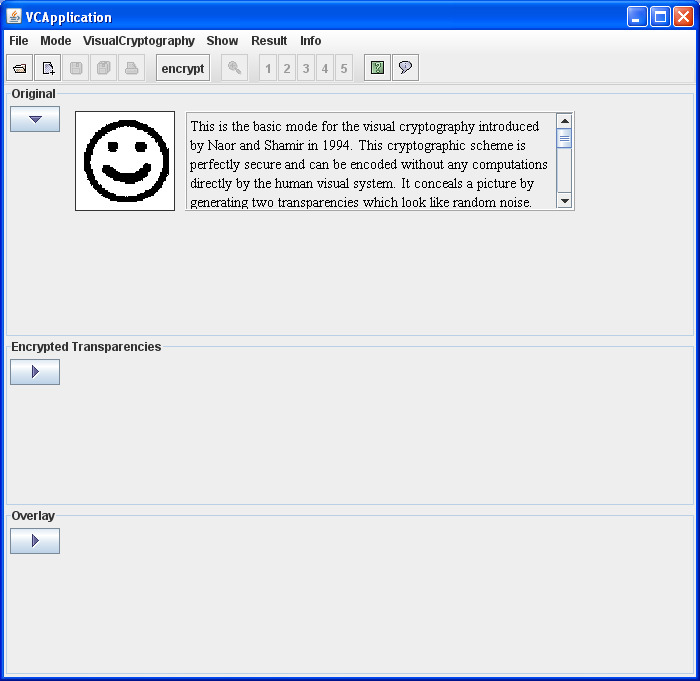
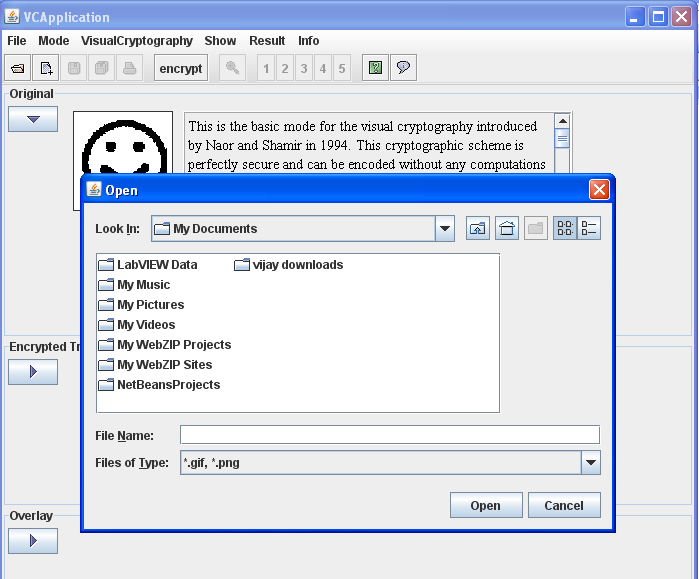


Fig: User interface which allows the users to work with Steganography too



To encrypt a image proceed with the following procedure:

* Select file menu
* Select load file sub menu
* Load .gif or .png formatted images

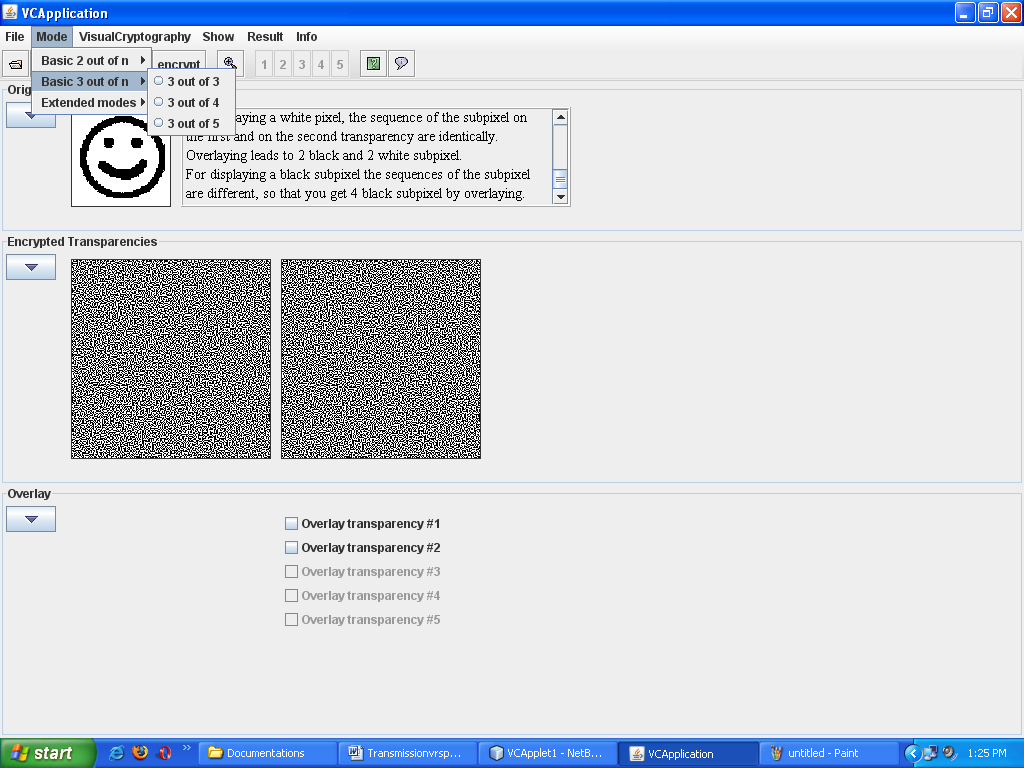


Fig: We can select mode of encryption by selecting Mode menu

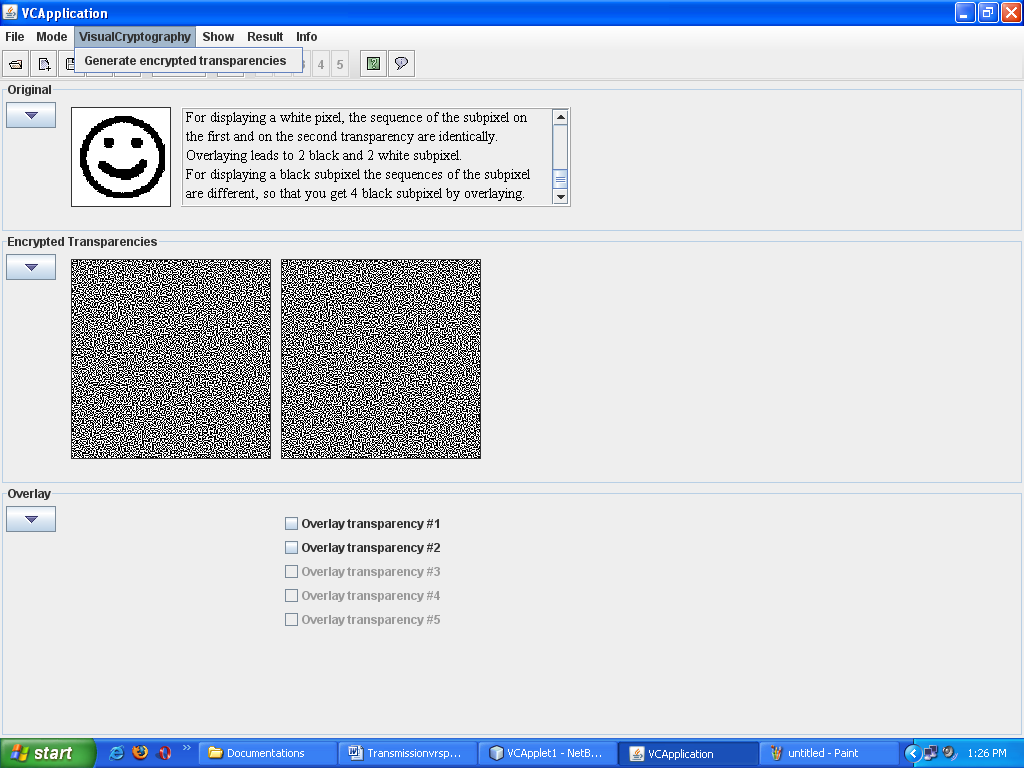
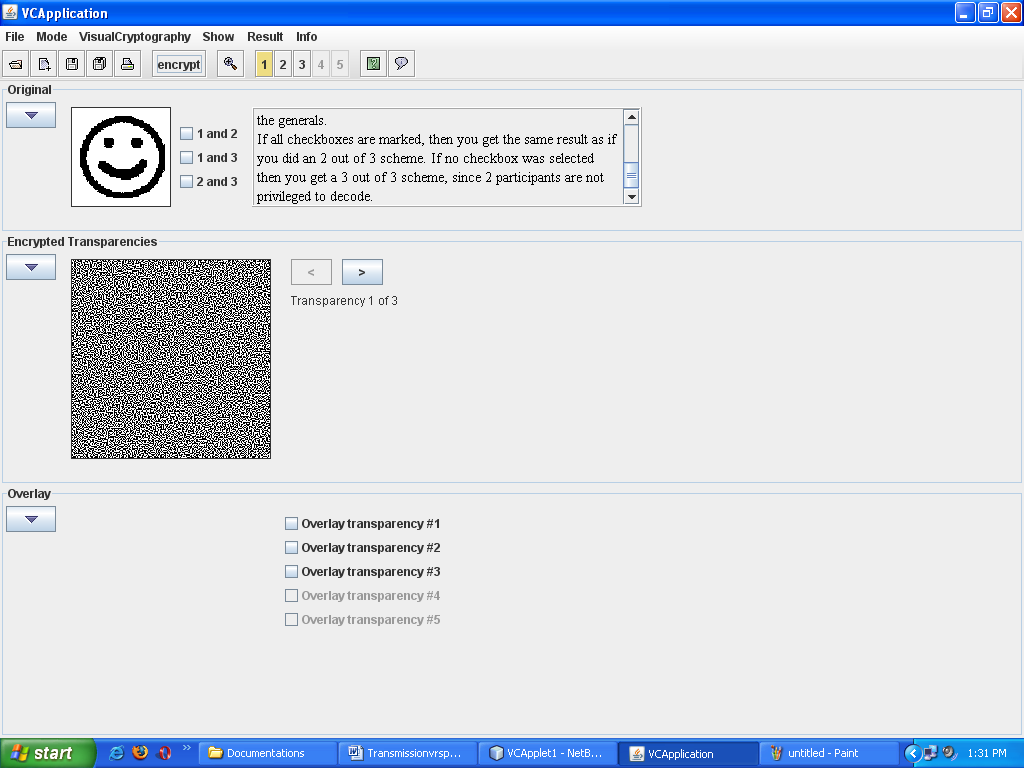


Fig: Generate encrypted transparencies submenu generates transparencies



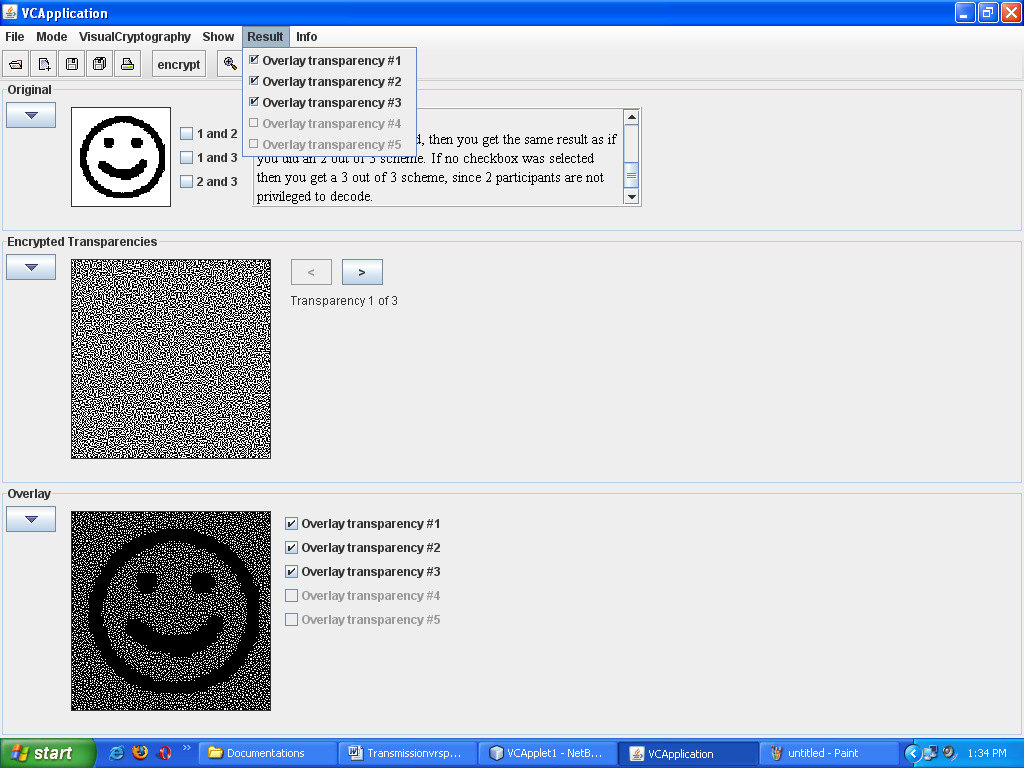


Fig: decrypted image

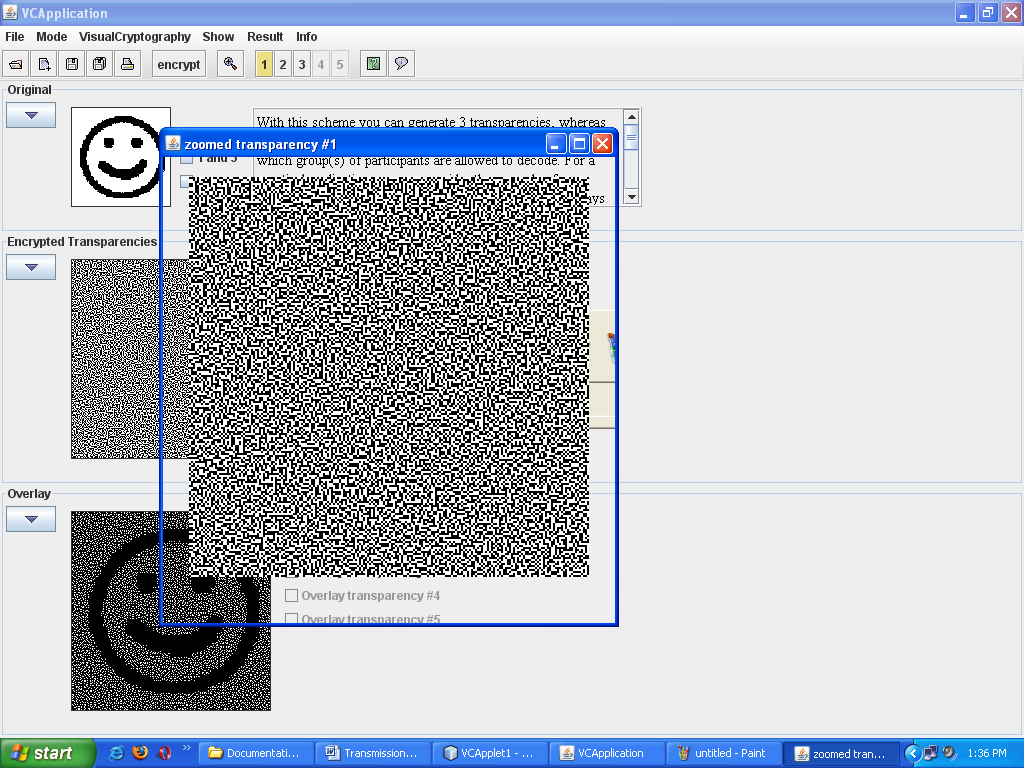


Fig: Zooming option supports zooming of transparencies

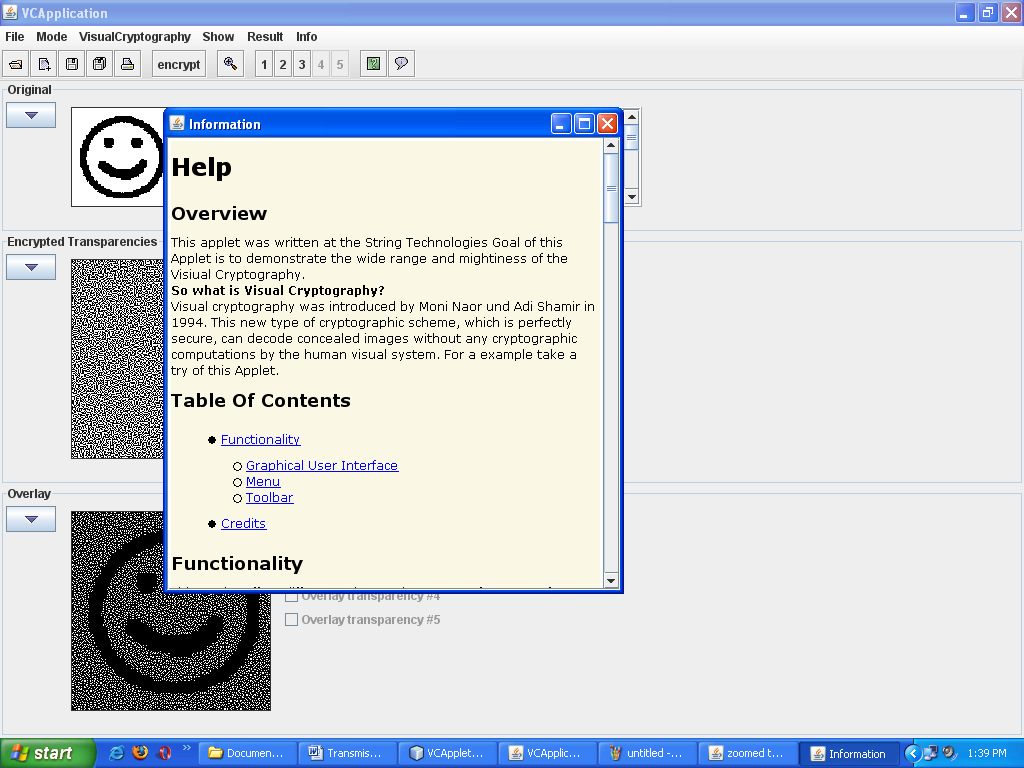


Fig: Help dialogue provides information about usage of this application

**CHAPTER 8: CONCLUSION**

Practical Training is a very important part of the curriculum as it strengthens the concepts and enhances knowledge about the practical implementation of all the theory concepts, we have learnt so far in different subjects. This project is used to keep a track on reserving the seat to the passenger and to provide facilities to complete this task. It helps managing the system very efficiently and conveniently.

This system is working properly and meets all the user requirements and subjected to future enhancements and modification as per user requirements. Although the project work has been done in a complete and detailed manner but due to the constraint of time, we could not include some more features we wanted to. We left these features as a part of the future development.

**CHAPTER** **9: BIBLIOGRAPHY**

**BOOKS**

* The Complete Reference (Java)  By Herbert Schildt
* Gray Cornett, Horstmann, CorelJava, Sunsoft Press, 1996.
* Patrick Naughton & Herbert Schildt, Java : The Complete Reference, Tata McGraw-Hill, Macrch 1997.
* Grady Booch, Object Oriented Analysis and Design with Applications, the Benjimin/Cummings, 1994.

**REFERENCE**

[1] A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[2] G. R. Blakley, “Safeguarding cryptographic keys,” in *Proc. National Computer Conf.*, 1979, vol. 48, pp. 313–317.

[3] M. Naor and A. Shamir, “Visual cryptography,” in *Proc. EUROCRYPT’ 94*, Berlin, Germany, 1995, vol. 950, pp. 1–12, Springer-Verlag, LNCS.

[4] M. Naor and B. Pinkas, “Visual authentication and identification,” in *Proc. CRYPTO’97*, 1997, vol. 1294, pp. 322–336, Springer-Verlag LNCS.

[5] T. H. Chen and D. S. Tsai, “Owner-customer right protection mechanism using a watermarking scheme and a watermarking protocol,” *Pattern Recognit.*, vol. 39, pp. 1530–1541, 2006.

[6] P. Tuyls, T. Kevenaar, G. J. Schrijen, T. Staring, and M. Van Dijk, “Security displays enabling secure communications,” in *Proc. First Int. Conf. Pervasive Computing, Boppard Germany, Springer-Verlag Berlin LNCS*, 2004, vol. 2802, pp. 271–284.

[7] C. Blundo, A. De Bonis, and A. De Santis, “Improved schemes for visual cryptography,” *Designs, Codes and Cryptography*, vol. 24, pp. 255–278, 2001.

[8] G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, “Visual cryptography for general access structures,” *Inf. Computat.*, vol. 129, pp. 86–106, 1996.

[9] N. K. Prakash and S. Govindaraju, “Visual secret sharing schemes for color images using halftoning,” in *Proc. Int. Conf. Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, 2007, vol. 3, pp. 174–178.

[10] H. Luo, F.X.Yu, J. S. Pan, and Z. M. Lu, “Robust and progressive color image visual secret sharing cooperated with data hiding,” in *Proc. 2008 Eighth Int. Conf. Intelligent Systems Design and Applications*, 2008, vol. 3, pp. 431–436.

[11] Y. C. Hou, “Visual cryptography for color images,” *Pattern Recognit.*vol. 1773, pp. 1–11, 2003.

[12] F. Liu, C. K. Wu, and X. J. Lin, “Color visual cryptography schemes,” *IET Inf. Security*, vol. 2, no. 4, pp. 151–165, 2008.

[13] S. J. Shyu, S. Y. Huang, Y. K. Lee,R. Z.Wang, andK.Chen, “Sharing multiple secrets in visual cryptography,” *Pattern Recognit.*, vol. 40, no. 12, pp. 3633–3651, 2007.

[14] P. A. Eisen and D. R. Stinson, “Threshold visual cryptography schemes with specified whiteness levels of reconstructed pixels,” *Designs, Codes and Cryptography*, vol. 25, pp. 15–61, 2002.

[15] S. Droste, “New results on visual cryptography,” in *Proc. CRYPTO’96*, 1996, vol. 1109, pp. 401–415, Springer-Verlag Berlin LNCS.

[16] G. Ateniese, C. Blundo, A. De Santis, and D. R. Stinson, “Extended capabilities for visual cryptography,” *ACM Theoretical Comput. Sci.*, vol. 250, no. 1–2, pp. 143–161, 2001.

[17] D. S. Wang, F. Yi, and X. B. Li, “On general construction for extended visual cryptography schemes,” *Pattern Recognit.*, vol. 42, pp. 3071–3082, 2009.