University of the Philippines Manila

College of Arts and Sciences

Department of Physical Sciences and Mathematics

# A Reversible Watermarking Java Application for Medical Images

A special problem in partial fulfillment

of the requirements for the degree of

Bachelor of Science in Computer Science

Dan M. Pantano

2008 - 25371

April 2013

## ACCEPTANCE SHEET

The Special Problem entitled “A Reversible Watermarking Java Application for Medical Images” prepared and submitted by Dan M. Pantano in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
**Vincent Peter C. Magboo, M.D., M.S**
Adviser

**EXAMINERS:**

| | **Approved** | **Disapproved** |
|---|---|---|
| 1. Gregorio B. Baes, Ph.D. (candidate) | _____ | _____ |
| 2. Avegail D. Carpio, M.S. | _____ | _____ |
| 3. Richard Bryann L. Chua, Ph.D. (candidate) | _____ | _____ |
| 4. Aldrich Colin K. Co, M.S. (candidate) | _____ | _____ |
| 5. Perlita E. Gasmen M.S. (candidate) | _____ | _____ |
| 6. Ma. Sheila A. Magboo, M.S | _____ | _____ |
| 7. Geoffrey A. Solano,Ph.D. (candidate) | _____ | _____ |
| 8. Bernie B. Terrado, M.S. (candidate) | _____ | _____ |

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

_____
**Avegail D. Carpio, M.S.**
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____
**Marcelina B. Lirazan, Ph.D.**
Chair
Department of Physical Sciences and Mathematics

_____
**Alex C. Gonzaga, Ph.D., Dr. Eng'g**
Dean
College of Arts and Sciences

# Abstract

Medical Images are widely used around the world today. Especially with new technology being developed, they are being used for more and more applications in the medical field. As such, the security of the transfer of medical images is a necessity. Possible situations might be unauthorized access, tampering of the images and even confusion of a medical image for a different patient. Usually, image watermarking is used to achieve this. However, inserting a watermark causes permanent distortions to the image used. For different applications, these are acceptable for as long as the image is visually comparable to the original. But for medical images, it is necessary that the image must exactly be the same as the original, as even small changes could cause a wrong diagnosis and possibly even loss of human life.

The Reversible Watermarking Application has been built to address these problems. Reversible Watermarking has the added capability of perfectly restoring the carrier image used. It allows the user to embed a watermark file to a single or multiple grayscale medical images. Furthermore, the user can add security by providing a password which is then used to generate a key for the encryption of the watermarked image using the Advanced Encryption Standard. The Reversible Watermarking algorithm used, which relies on Congruence Computations, allows for a high embedding capacity with a relatively small complexity when compared with other reversible watermarking algorithms. Being the standard used by the USA, the Advanced Encryption Standard assures that the encrypted image is secure against possible attacks to gain access to the image.



*Keywords:* Reversible Watermarking, Congruence Computations, Medical Images

Contents

# I. Introduction

## A. Background of the Study

Medical images are now widely used in the medical practice of countries all over the world. Nowadays especially, technology has increased the applications in which they are being used. Doctors and medical personnel use these images in order to correctly diagnose the condition of a patient in order to determine the proper treatment procedure to be administered.

As such, it is important that the transfer of medical images be made secure from possible unauthorized access and tampering. Also, it should be assured that the information being processed and examined should belong to the correct patient. A more commonly used method to do this is the process of Image Watermarking. However, inserting digital watermarks in medical images permanently distorts the signal of the host image in order to complete the watermarking process.[1] On other applications, the loss of image fidelity is not prohibitive as long as the original and watermarked images are perceptually equivalent. However, this is not the case for medical images, where small distortions in the patient's diagnostic image may cause errors in the diagnosis and treatment with life threatening consequences. So there is a need to retrieve the original image from the watermarked one.[2] This is where Reversible Image Watermarking comes in.

Reversible Image Watermarking, aside from providing a method of inserting a watermark into an image, also provides a method of retrieving the original image along with

the watermark used from the watermarked result.[3] This assures that the image has not been tampered with, and has been transferred securely. The retrieved watermark would also serve as authenticator for the sender and for the patient. Such a method would help meet the need for securing medical images and assuring that the same images will be unchanged upon examination.

A reversible watermarking scheme was introduced by Coltuc and then improved by Chaumont and Puech.[4] The scheme relies on the transformation of image pixels which satisfy some constraints, and then embedding of the data on these pixels through simple additions. The transform performed on the selected pixels then induces a congruence equation which are then checked during the extraction process in order to identify pixels which contain the watermark data.

Data encryption is defined as: "the process of converting information into an encrypted form, so that it is intelligible only to someone who knows how to decrypt it to obtain the original message".[5] It involves using a plaintext(original data) and then using an encryption algorithm and key in order to convert it into ciphertext. It is used to protect data such that only an authorized entity will be able to access it. As of 2001, the United States National Institute of Standards and Technology has declared a standard for their file encryption needs, the Advanced Encryption Standard(AES).[6] As of now, the AES encryption standard is still widely used, and the security of the encryption algorithm is still assured. All attacks in order to crack it has failed, or will take a very large amount of time which is impossible with current technologies to complete. [7]

B. Statement of the Problem

Traditional Image Watermarking has been a very useful tool for the security of images and other media. However, such methods produce a permanent distortion on the original image in order to complete the watermarking process. Such a distortion could be crucial in the medical profession and specifically in the use of medical images, where each image is examined so thoroughly that a small change might have a drastic effect on the diagnosis and could lead to possible endangerment of human life.

At certain situations on the usage of medical images, it is necessary to limit the people who are given access to the image. Thus, securing it depends on two situations: First, it should be assured that only entitled users will have access to the information, and second, there should be proof that the data are exactly as sent by the authorized user.

With the increasing use of medical images in the industry, it is possible that there might be confusion when multiple images are transferred to different parties concerned. Therefore a confirmation should also be provided that the image sent does indeed belong to the correct patient.

Hence there is a need for a Reversible Watermarking method that not only provides methods for securing medical images, but also allows the original image and the watermark to be extracted in order to assure correct diagnosis. Additionally, the watermark to be used may be a file which contains the patient information. In order to avoid unauthorized access, the encryption process uses a key. The extracted watermark

can then be used for the purpose of authenticating the identity of the sender and as well as so to make sure that the data belongs to the correct patient.

C. Objectives

To create a Reversible Watermarking Application with the following capabilities:

- Use Congruence Computations for the Watermark Embedding and Extraction Process.
- Provide an option to watermark images of either .jpg, .png, or .bmp file format and return it as a .png image.
- Provide an option to watermark single or multiple images.
- Provide an option for either a .jpg, .bmp, .png, or a .txt file type for the watermark to be used.
- Provide a function to encrypt the image using the Advanced Encryption Standard using a pass phrase to generate the key.
- Provide an option to extract the watermark and original image from a single or multiple images.
- Provide options to save the watermarked image on a different folder and use a different filename.
- Provide an error notification on the embedding process if there is not sufficient space to embed the watermark.
- Provide an error notification on the extraction process if the original image and watermark cannot be retrieved.

D. Significance of the Study

The application will improve the usage of medical images by providing a method that assures the security and correctness of the images used. The watermarking procedure prevents unauthorized access of the image and makes sure that it has not been tampered with. The retrieval of the original image makes sure that any change obtained from watermarking will be removed and will possibly not cause a wrong diagnosis by the medical personnel in charge. The watermark which will be retrieved as well can serve the purpose of authenticating the user and could also be used to carry information about the patient, thereby making sure that the image used belongs to the correct person.

The algorithm to be used, which relies on congruence computations, provide a high embedding capacity for the watermark, which does not require image compression in order for the embedding process to succeed. Moreover, compared to other reversible watermarking schemes, the algorithm has a smaller complexity. The reversible watermarking scheme also does not increase the file size of the image when after going through the embedding process. [4]

Using the Advanced Encryption Standard(AES) makes sure that the medical image is secured, as it is the current data standard as defined by the United States National Institute on Standards and Technologogy.[6] The AES has been deemed as secure, and all attacks on it have failed or will take an impossible amount of time to succeed.[7]

The usage of the Portable Network Graphics(.png) file format for the watermarked

image allows for the image quality to be retained due to it being a lossless image format. The .png format is suitable for medical images where precise details of the image must be retained.[8]

E. Scope and Limitations of the Study

1. The application will only make use of the Congruence Computations Method for the process of Reversible Watermarking.
2. The reversible watermarking application will return images of file type PNG.
3. The application will use the Advanced Encryption Standard Algorithm(AES) for File Encryption purposes.
4. The application will only be used for watermarking and image retrieval operations.
5. The watermark to be used must be of smaller file size than the carrier image.
6. The application will only be used for static type of images.

F. Assumptions

1. The reversible watermarking application will only be used for grayscale images, as this model is the one most commonly used for medical images.[9]
2. The images to be used is assumed to have already been optimized before processing.
3. The transfer of the images and the pass phrase to be used does not concern the application.

## II. Review of Related Literature

The use of medical images in the industry has been increasing in the recent times. As such, it is vital that these images used be free from any tampering and the transfer of these must be always secure. The security is based on the following criteria[2]:

(i)Confidentiality : The protection of data from unauthorized disclosure.

(ii)Integrity : The assurance that data received is exactly as sent by an authorized entity.

(iii)Authentication : The assurance that the communicating entity is the one that it claims to be, and proof that the information belongs to the correct patient.

To help with the security of medical images, the use of watermarks might be a possible solution. However, traditional image watermarking procedures produce permanent distortions to the original image, and for the field of medicine, even small distortions are simply not acceptable, because these changes might mean the difference between a correct or a wrong diagnosis. Thus, after watermarking, there is a need to retrieve the watermark, as well as the original image. The process of reversible watermarking satisfies this requirement and is viewed as a solution in the problem of securing medical images.[3]

Reversible Watermarking, in contrast from traditional Watermarking methods, provides a way for the user to retrieve the original image from the watermarked. The retrieved watermark can then be used for authentication.[10] Reversible Watermarking schemes can be

divided into two categories. The fragile and semi fragile. Fragile techniques lose the watermark once a slight modification has been made, signifying that the data has been compromised, while semi fragile techniques are able to withstand some slight changes to the image such as compression. Robust techniques, or those that can withstand some changes such as geometric operations(scaling, translation, etc.) can be classified under the semi fragile category.[11]

Reversible watermarking methods in the past often compress the original values of the embedding area in order to provide space for the digital watermark.Tian has proposed the method of reversible watermarking using the difference expansion that allows for high payload capacity. It does so by exploring the redundancy in the digital content to achieve reversibility.[12]

Tian's reversible watermarking scheme has then been improved by Kallel et al.[13]. They do so by first dividing the image into blocks and computes separate watermarks corresponding to each line or column. The proposed scheme is fragile, and detects if the image has been modified, with the additional capability of localizing where the modification has been made.

Another study has been done using a modified difference expansion by Yaqub et al. Their proposed reversible watermarking scheme is comparable to that of Tian, Alattar and Celik.[14] The scheme has a higher complexity with more calculations needed, but it justifies this by providing a higher payload capacity.

A reversible visible watermarking method was presented by Yang et al.[15] The scheme presented considers Human Visual System(HVS) characteristics in order to preserve the visual quality of the image. In order to achieve reversibility, a reconstruction/recovery packet, which is utilized to restore the watermarked area, is reversibly inserted into a non-visibly-watermarked region.

Another proposed method was done by Liew et al. The method divides the image into Regions of Interest (ROI) and Regions of Non Interest (RONI). The watermark data from the ROI's are then stored in the RONI's. The method also employs Run-Length Encoding, a type of lossless data compression scheme which is used in order to allow higher embedding capacity.[16]

The previous schemes discussed here fall under the fragile classification. The scheme proposed by Narawade and Kanphade on the other hand, is a robust one.[17] The reversible watermarking method presented is based on the Discrete Cosine Transform(DCT) and is resistant to scaling and translation of the image. However, it is not resistant to rotation, and is not fully robust.

Encryption has been used before in a proposed reversible watermarking scheme by Golpira and Danyali.[18] The method uses the algorithm of histogram shifting in the wavelet domain in order to embed the watermark. The watermark embedded is first encrypted with a private key in order to ensure security. It also uses thresholds in order to decrease distortion in the watermarked image and improve the image quality.

# III. Theoretical Framework

## A. Reversible Watermarking

A digital watermark is a symbol or marker embedded on an image which is often used to identify ownership of the copyright of the material. Watermarking is the process of embedding the watermark data into the image itself.[19] It is also sometimes referred to as data embedding and information hiding.

The process of watermarking inevitably produces some distortion to the image used. As much as possible, this distortion is made to be as small as possible while meeting other qualifications, such as robustness and sufficient capacity for the payload. These distortions, no matter how small, are permanent and irreversible.[20] However, there are some cases where even a small change in pixel values cannot be allowed, such as in military or medical use of images. Therefore traditional watermarking is not sufficient for such situations, and they are when reversible watermarking is applicable.[21]

Reversible watermarks, like traditional ones, can be used for content authentication. But it also provides the additional advantage that when the original image is needed, one can remove the watermark to retrieve the unwatermarked content. Figure 1 shows the difference of traditional with reversible watermarking.

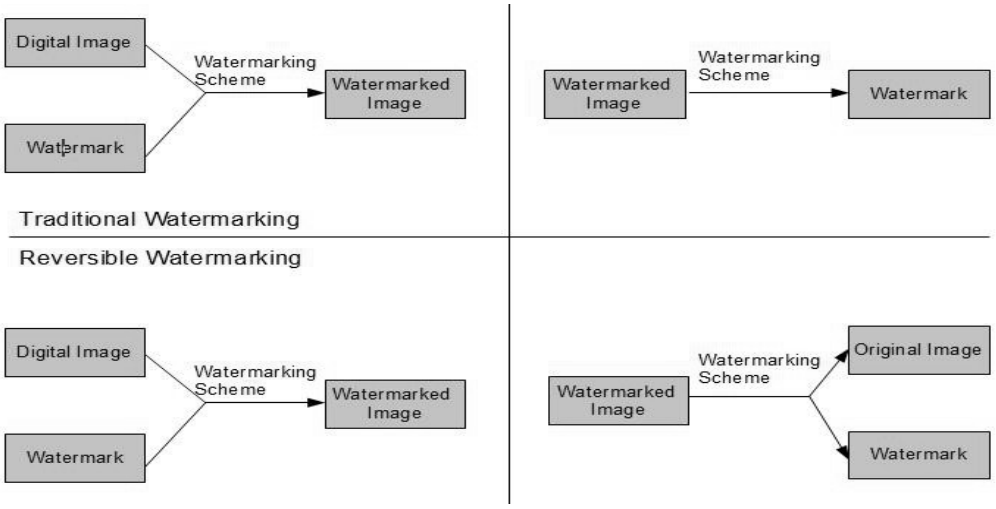


Figure 1: Traditional vs Reversible Watermarking

Reversible watermarking schemes are divided into two classifications, the fragile and the semi fragile.[11]

Fragile algorithms work in such a way that once the watermarked image has been altered in any way, the embedded watermark cannot be retrieved anymore, and the original image cannot be recovered as well.

Semi-fragile algorithms on the other hand, can present a certain degree of robustness when a specific operation is applied to the watermarked image such as scaling, rotation, etc. The watermark along with the original content is still recoverable.

A special type of reversible watermarking scheme that can be classified under the semi-fragile is the robust algorithm. These schemes are employed when the watermarked image must withstand any operation done on it and must still allow the watermark and the original image to be retrieved.

B. Reversible Watermarking using Congruence Computations

The reversible watermarking algorithm to be used was proposed by Chaumont and Puech and relies on congruence computations to achieve reversibility. The algorithm proposed has a high embedding capacity along with a small complexity. The visual quality of the watermarked image is reduced, but since the image will be restored to the original after watermark extraction, it is not considered to be a problem.[4]

The algorithm first defines a constant integer value *n* that is greater than or equal to 3(for this application, n is given as 4), and the T transform which takes on two integers $x_1$ and $x_2$ as input and returns an integer:

$$T(x_1, x_2) = (n + 1).x_1 - n.x_2$$

For an image *I* with size *N*(*N* = M x N pixels) whose gray-levels belong to [0, L] where L = $2^B$ and B is the number of grey levels of the image. Each pixel *i* is then classified into three possible states:

- Embedding: A pixel which contains a portion of the watermark.
- To-correct: A pixel which has been modified but does not embed any portion of the watermark. These pixels will be corrected in the extraction process.
- Original: A pixel whose value is unchanged.

A pixel *i* is said to be in the embedding state if it satisfies the condition:

$$0 <= T(I(i), I(i + 1)) \text{ and } T(I(i), I(i + 1)) + n <= L$$

All pixels *i* which are in the embedding state:

1. Are transformed such that $I_T(i) = T(I(i), I(i+1))$ with $I_T(i)$ as the resulting pixel,

2. Must then embed a coeffiecient *w* which belongs to [1, *n*] such that

$$I_w(i) = I_T(i) + w \text{ and}$$

$$I_w(i) = (n+1).I(i) - n.I(i+1) + w, \text{ such that } w \neq 0$$

From that, we can obtain the original pixel *I(i) as:*

$$I(i) = \frac{I_w(i) + n.I(i+1) - w}{n+1}$$

and we obtain the property that:

$$(I_w(i) + n.I(i+1)) \ mod \ (n+1) \neq 0.$$

This congruence property obtained will then be used to detect an embedding pixel during the extraction process.

A pixel *i* is said to be in the to-correct state if it satisfies the negation of the condition for the embedding state. That is:

$$0 > T(I(i), I(i+1)) \text{ and } T(I(i), I(i+1)) + n > L$$

All pixels in the to-correct state is then modified such that:

$$c \leftarrow (I(i) + n.I(i+1)) \ mod \ (n+1);$$
$$if \ (I(i) - c) < 0 \ \ then \ \ c \leftarrow -(n+1-c);$$
$$I_w(i) \leftarrow I(i) - c.$$

where *c* is defined as a corrective code, and are embedded (into the embedding pixels) along with the watermark in order to achieve the reversibility of the to-correct pixels.

After the modification done by computing for c, the pixel $I_w(i)$ now checks the property:

$$(I_w(i) + n.I(i+1)) \ mod \ (n+1) = 0.$$

This congruence property then allows the detection of a to-correct pixel during the extraction process.

For a top-bottom-left-right image scan order, an original pixel **must** always be present just before a to-correct pixel.

The embedding process is now composed of the following steps:

1. Classify each pixel according to the conditions.
2. Embed into the embedding pixels the watermark along with the corrective codes obtained.

For the extraction process, the image scan order is inverted, and it is composed of the following steps:

1. Extract the watermark and corrective codes from the embedding pixels, and revert them to the original. Obtain a list of all the to-correct pixels.
2. Separate the watermark from the corrective codes, and correct the to correct pixels obtained in step 1.

The whole algorithm for the reversible watermarking method is as follows:

```
define int n, c, k, t;
Function Embed(Image I, Watermark m)
        define list embedding;
        define list ccodes;

        for i from 1 to N-1
                t = T(I(i), I(i+1));
                if(0<=t) && (t + n <= L)
                then
                        Iw(i) = t;
                        embedding.add(i);
                else
                        next = look_for_next_embedding_pixel(i);

                if(next-i is odd)
                then
                        k = i-1;
                        Iw(i) = I(i-1);
                        embedding.remove(i);
                else
                        k = i;
                endif

                for j from k to next-1          //alternate original and to-correct pixels
                        if(j.tocorrect())
                        then
                                c = (I(j) + n.I(j+1))%(n+1);
                                if((I(j)-c<)0)
                                then
                                        c = n + 1 - c;
```

```
                    endif
                    Iw(j) = I(j) - c;
                    ccodes.add(j);
                endif
            endfor
            i = next -1
            endif
        endfor

        generate sequence w = m + ccodes;
        for(all i in embedding)
            Iw(i) = Iw(i)  + w(i)          //w(i) is an element of [1, n]
        endfor
        return Iw
end
Function Extract(Image Iw)
        define list to_correct;
        define int v;
        define sequence w;                 //retrieved sequence
        I = Iw;                            //copy watermarked to original

        for i from N-1 to 1                //inverted order of scan
            v = (Iw(i) + n.I(i + 1)) % (n+1);

            if(v=0)
            then
                to_correct.add(i)
                i = i - 1;                 //next pixel is in original state
            else
                w = w + v;
                Iw(i) = Iw(i) - v;
```

```
                    I(i) = [Iw(i) + n.I(i+1)] / (n+1)        //inverted T function
            endif
        endfor

        separate from w the watermark m and corrective codes ccodes;

        for(all i in to_correct)
            I(i) = Iw(i) + ccodes(i);
        endfor

        return watermark file m
        return Image I
end
```

C. Encryption

Encryption is the process of converting information into an encrypted form, so that it is intelligible only to someone who knows how to decrypt it in order to obtain the original message.[5] It involves taking data that is called the "plaintext" and converting it to a "ciphertext" using an encryption algorithm and a key.

Encryption is most often used to protect information so that only an authorized entity can access it. Some applications include:

- Personal Use: using data encryption products such as the Pretty Good Privacy or PGP application in order to protect one's personal data.
- Securing Networks: enables secure communications and user authentication over open, unsecured networks like the Internet.
- Access Control: an example is in digital television, providers control access by encrypting audio and video signals. Subscribers are given a descrambling device, which contains the algorithm and key, in order to decrypt the pictures and sound.

D. Advanced Encryption Standard(AES)

The Advanced Encryption Standard Algorithm or AES is a specification for the encryption of electronic data that has been been used as a standard in the United States since 2001 as U.S. FIPS 197.[6] It was originally called "Rijndael" a combination of the names of the two Belgian cryptographers who developed it, Joan Daemen and Vincent Rijmen.

The AES processes sequences called blocks each with length of 128 bits, along with a key size of either 128, 192, or 256 bits. This key size determines the number of times the plaintext(data to be encrypted) undergoes transformation in order to produce the final output. These are 10, 12 and 14 cycles of repetition for keys of size 128, 192 and 256 bits respectively.

The algorithm used by the AES is classified as  a symetric-key algorithm. That is, the same key is used for both encyption and decryption of the data.

As for the security considerations on the safety of the AES for use, it is considered as safe from brute-force attacks due to the incredibly large amount of time needed to crack the algorithm. However, some attacks have cracked the AES with a slightly improved time from the brute force attack. The said attack however, would still take billions of years of computer time, and thus the algorithm is still considered as secure for now.[7]

E. Medical Imaging

According to the United States Food and Drug Administration[22],Medical Imaging refers to the several different technologies that are used to view the human body in order to diagnose, monitor, or treat medical conditions. Each type of technology gives different information about the area of the body being studied or treated, related to possible disease, injury, or the effectiveness of medical treatment.

Some of the areas under medical imaging include[23]:

- Computed Tomography: Commonly referred to as a CAT Scan, combines multiple X-ray images taken from different angles to produce detailed cross sections of the areas inside the body.
- Magnetic Resonance Imaging(MRI): A technology that uses radio waves and a magnetic field to create detailed images of organs and tissues.
- Positron Emission Tomography(PET): A type of nuclear medicine that provides physicians with information about how tissues and organs are functioning. It is often used in combination with the CT Scan.
- Ultrasound: Also known as medical sonography or ultrasonography, uses high frequency sound waves to create images of the inside of the body.
- X-ray: The oldest and most commonly used form of medical imaging. They use ionizing radiations to produce images of a person's internal structure by sending x-ray beams through the body, which are absorbed in different amounts depending on the density of the material.

# IV. System Design and Implementation

## A. Context Diagram

The Reversible Watermarking application will have only one type of user. The user will provide the carrier image/s, watermark file, and an optional encryption key then the application will return the watermarked image/s. In the same way, the user can provide the watermarked image/s and an optional decryption key and the application will return the watermark file and the original image/s. The context diagram is shown in Figure 2.

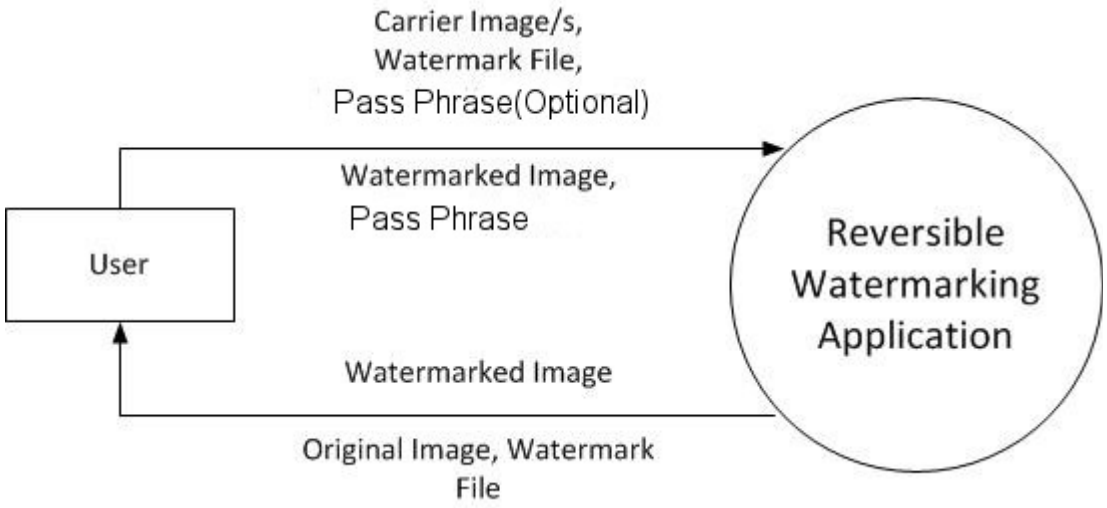


Figure 2: Context Diagram, Reversible Watermarking Application

B. Use Case Diagram

Upon starting the application, the user can either embed a watermark, extract an image and watermark, or to configure the options of the application. Upon choosing to embed a watermark, the user can select a carrier image(or images), select a watermark to be used, or provide an encryption key for the image(optional). When the extract image and watermark option is selected, the user can select an image(or images) that will undergo the extraction process, and provide a key to decrypt the image(optional). Finally, if a user selects to configure the options of the application, he can choose to save the folder where the outputs of the application are to be saved, or change the filename of the watermarked image, the extracted watermark, or the extracted image. The use case diagram for the application is illustrated in Figure 3.

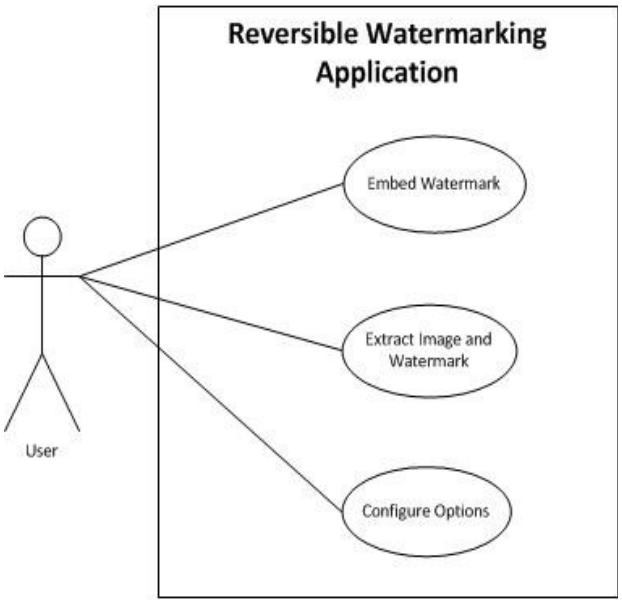


Figure 3: Use Case Diagram, Reversible Watermarking Application

Figures 4 and 5 show the Use Case Diagram if the User selects the "Embed Watermark" option and the Use Case Diagram if the user selects the "Extract Image and Watermark" option, while Figure 6 illustrates the use case diagram for the "Configure Options" menu.

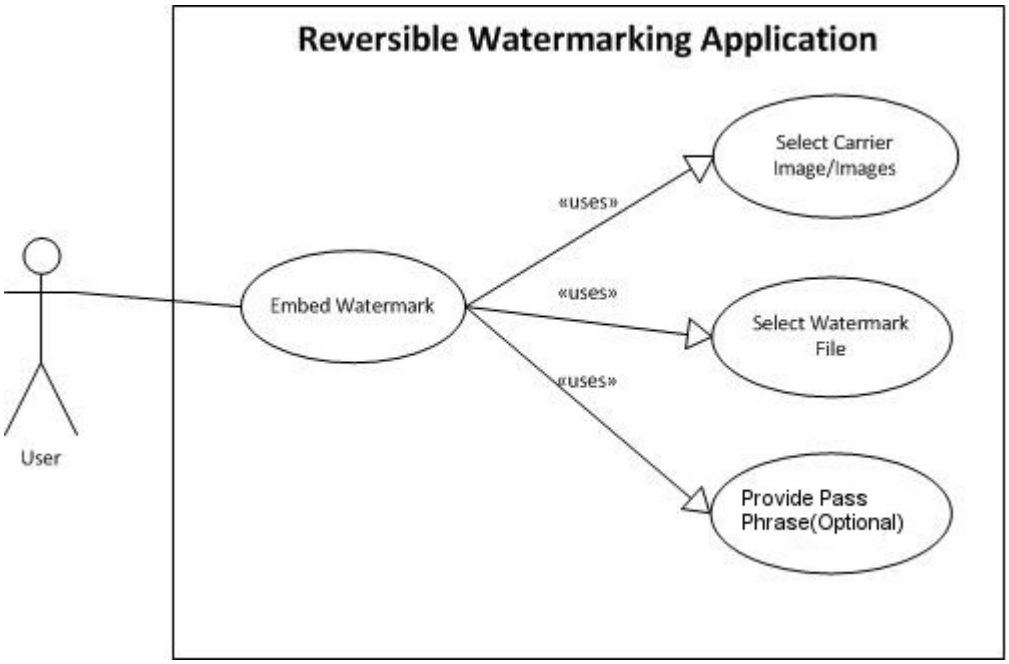


Figure 4: Use Case Diagram for the Embed Watermark Option, Reversible Watermarking Application

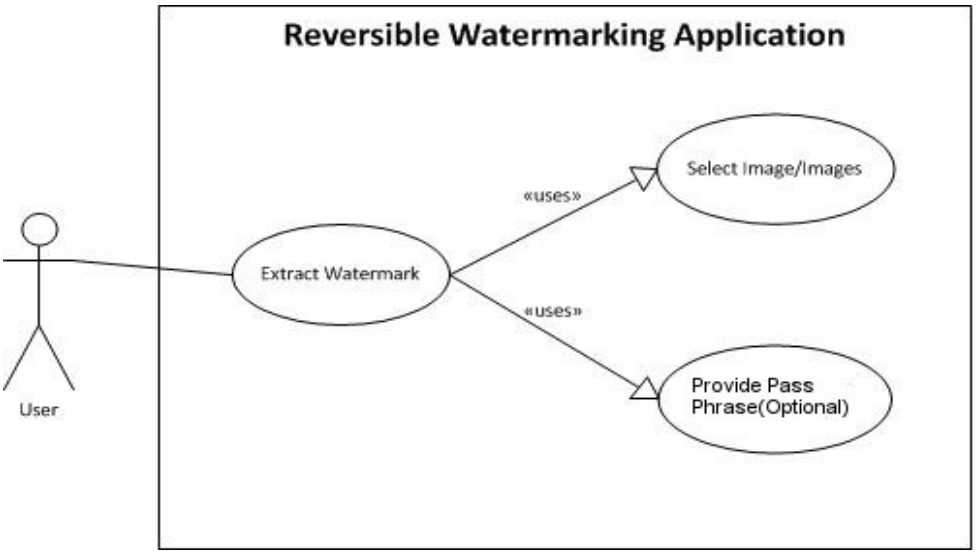


Figure 5: Use Case Diagram for Extract Image and Watermark Option, Reversible Watermarking Application

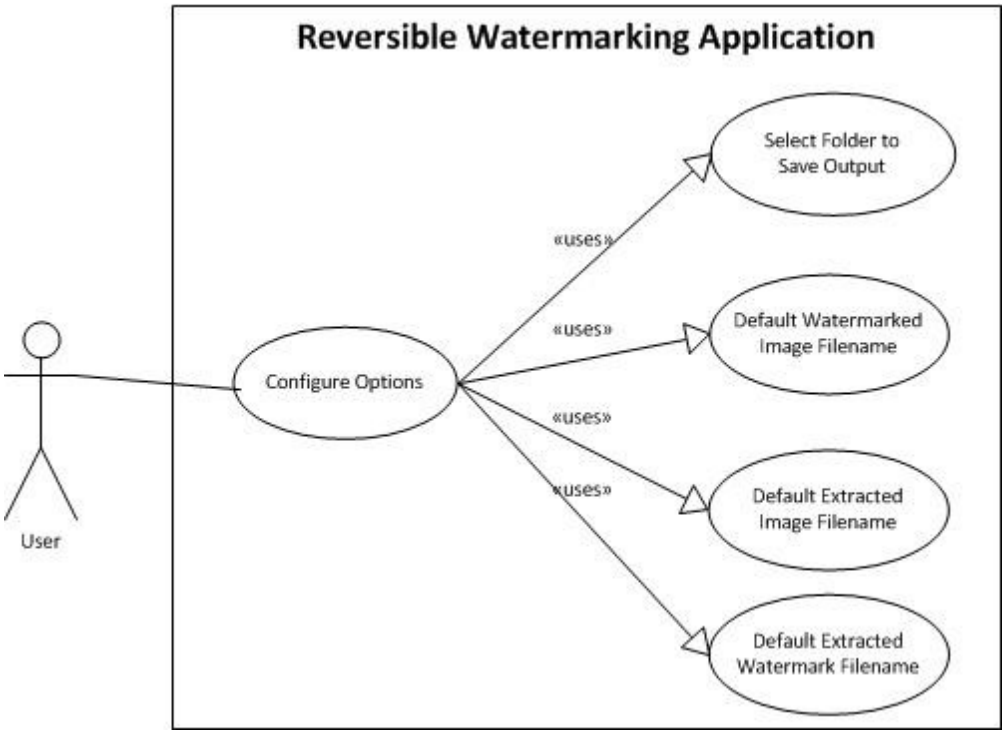


Figure 6: Use Case Diagram for Configure Options Menu, Reversible Watermarking Application

C. Class Diagram

The application will have three main classes, the RevWatermarkingApp class that uses the AppEmbedder and AppExtractor classes in order to perform the main functions of the application. The Class Diagram is illustrated in Figure 7.

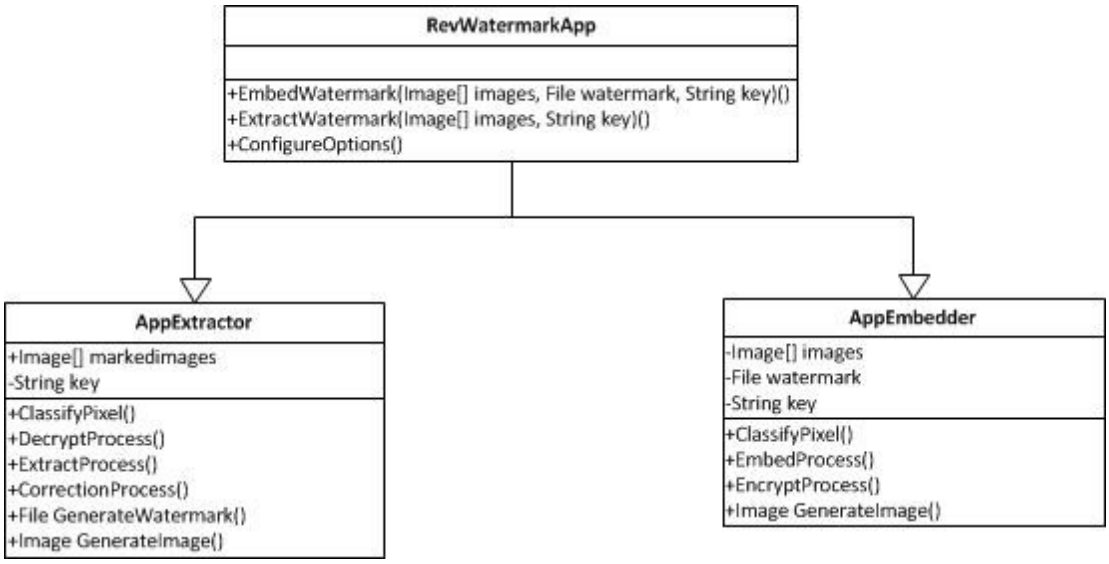


Figure 7: Class Diagram, Reversible Watermarking Application.

D. Flowchart

The flowchart for the application's embedding algorithm is presented in Figure 8.

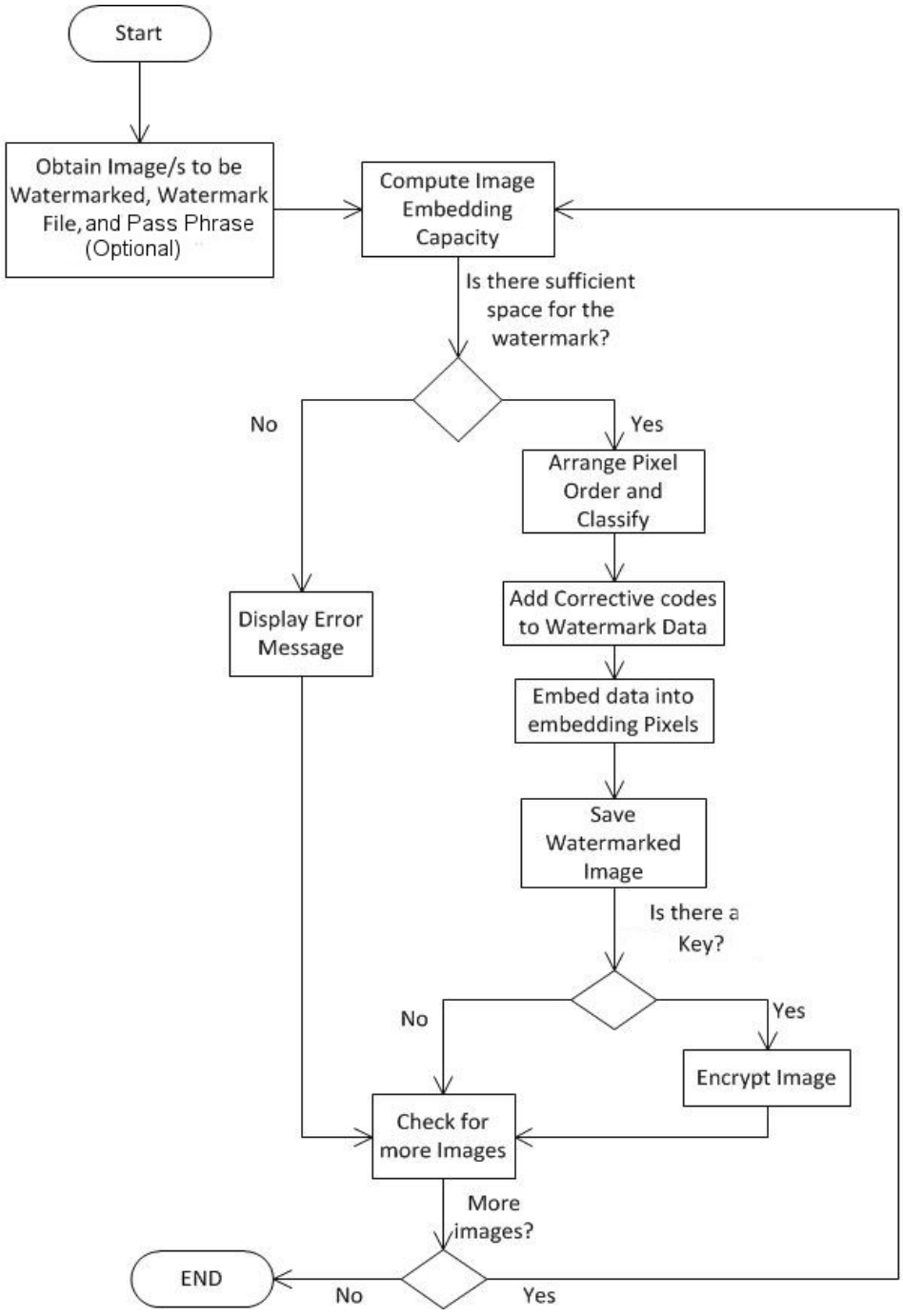


Figure 8: Flowchart for Embedding Algorithm, Reversible Watermarking Application

The flowchart for the application's extraction algorithm is presented in Figure 9.

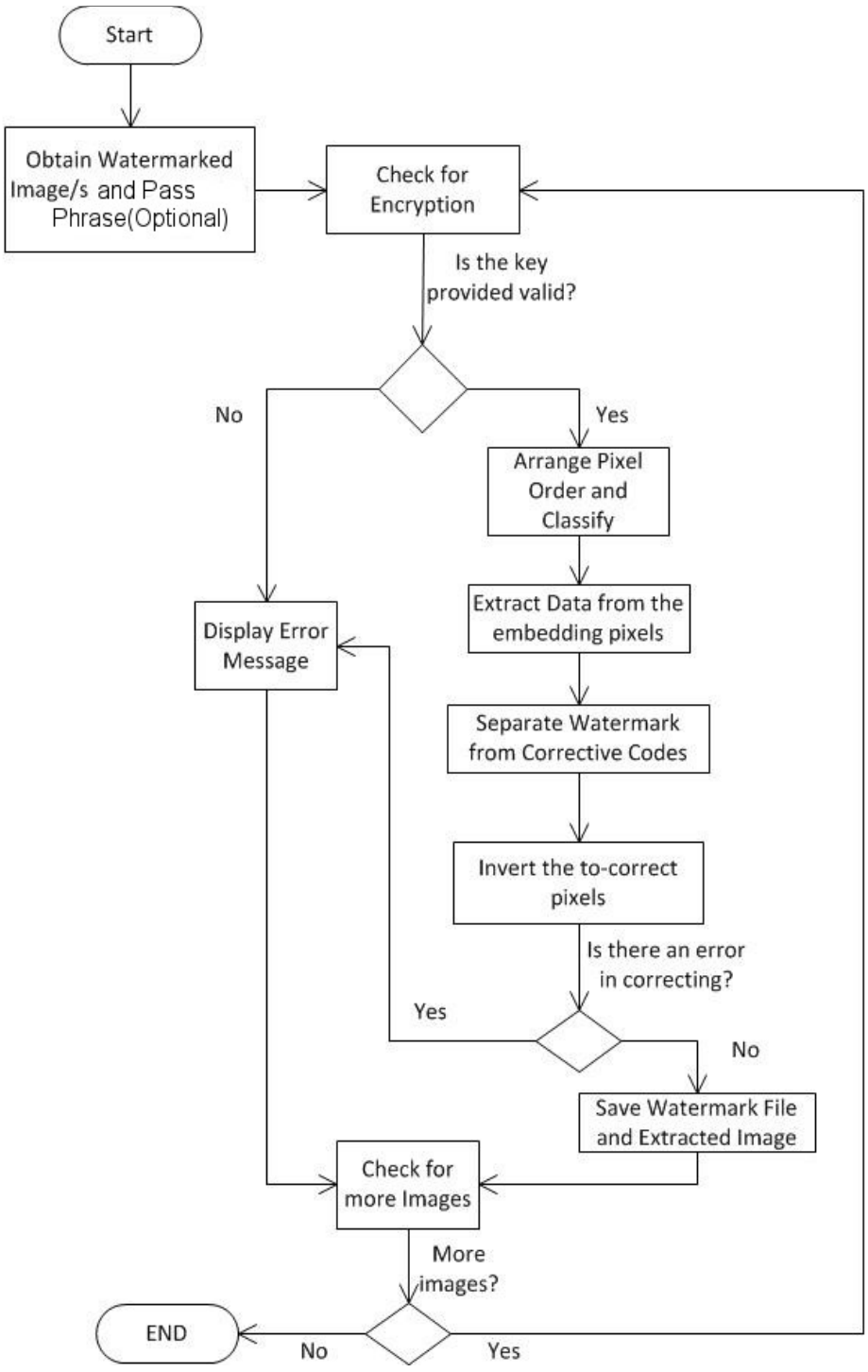


Figure 9: Flowchart for Extraction Algorithm, Reversible Watermarking Application

E. System Requirements

The application will have the following system requirements:

1. At least JRE or JDK 6 is installed

2. Operating System: Windows XP, Windows Vista, or Windows 7.

3. At least 2GB of RAM.

4. Processor: Intel Pentium Core 2 Duo Processor or Better.

# V. Results

The welcome page for the Reversible Watermarking Application can be seen in Figure 10. The welcome screen shows the different functionalities of the application. The user can then select a functionality by clicking on it and the application displays the selected page.


Figure 10: Welcome Page, Reversible Watermarking Application

If the user selects the Instructions icon, the user is redirected to the instructions page of the application. It contains information on how to use the watermark embedding and extraction functions of the application as seen in Figure 11.


Figure 11: Instructions Page, Reversible Watermarking Application

If the user selects the View Help Topics icon, the user is redirected to the Help Topics page. It contains three topics about the application, namely Reversible Watermarking, Congruence Computations and Encryption using the Advanced Encryption Standard. The user can then select a topic and a separate window pops up containing more information about the selected topic. Figures 12 to 15 show the Help Topics page, the help pages for the Reversible Watermarking, Congruence Computations and Encryption using the AES topics respectively.

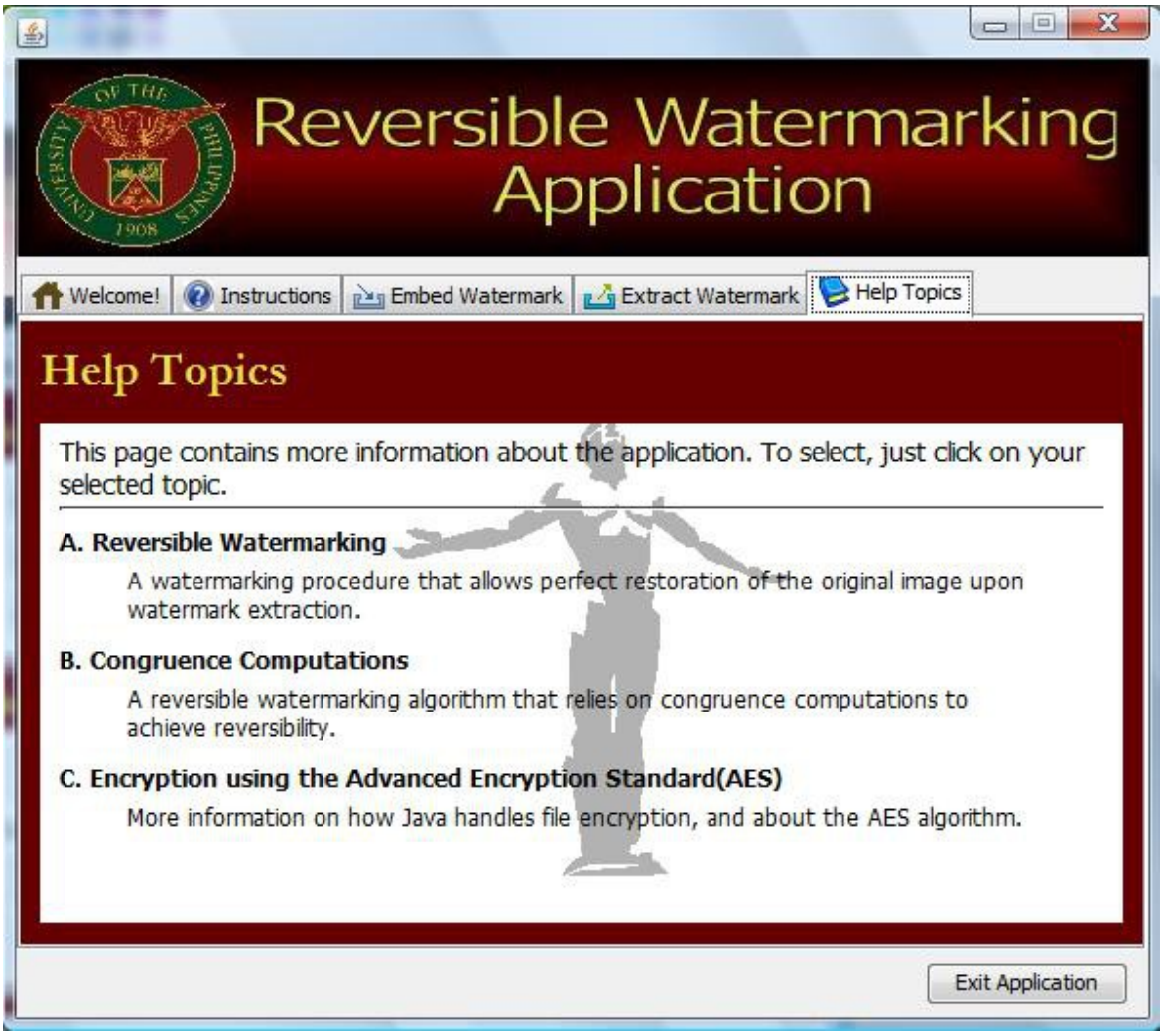


Figure 12: Help Page, Reversible Watermarking Application

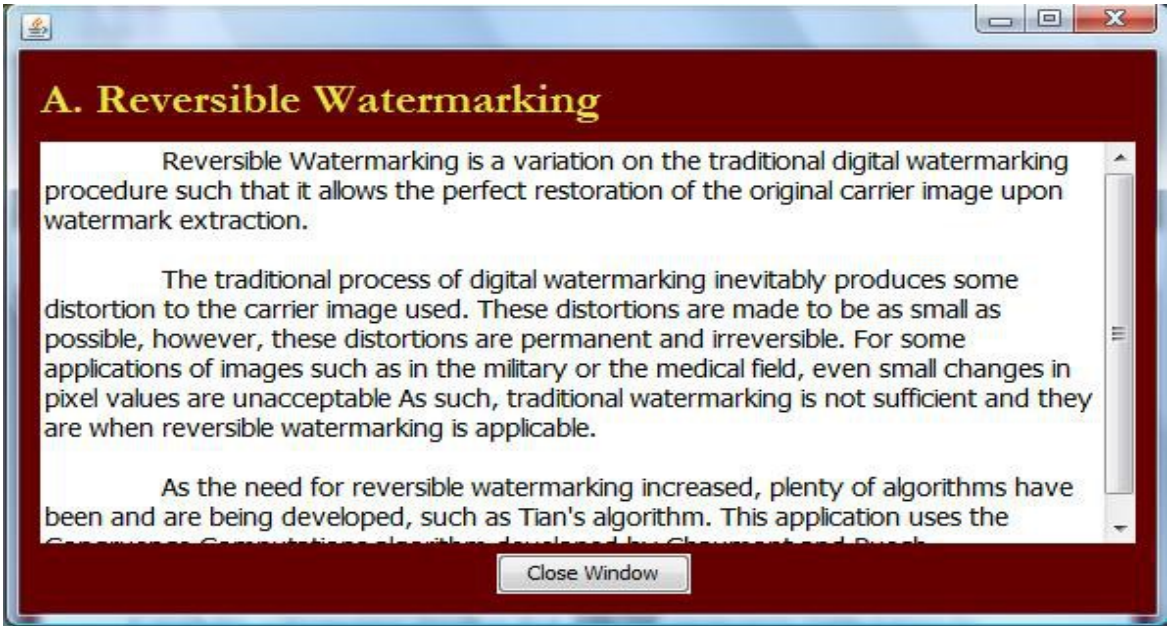


Figure 13: Reversible Watermarking Help Topic, Reversible Watermarking Application

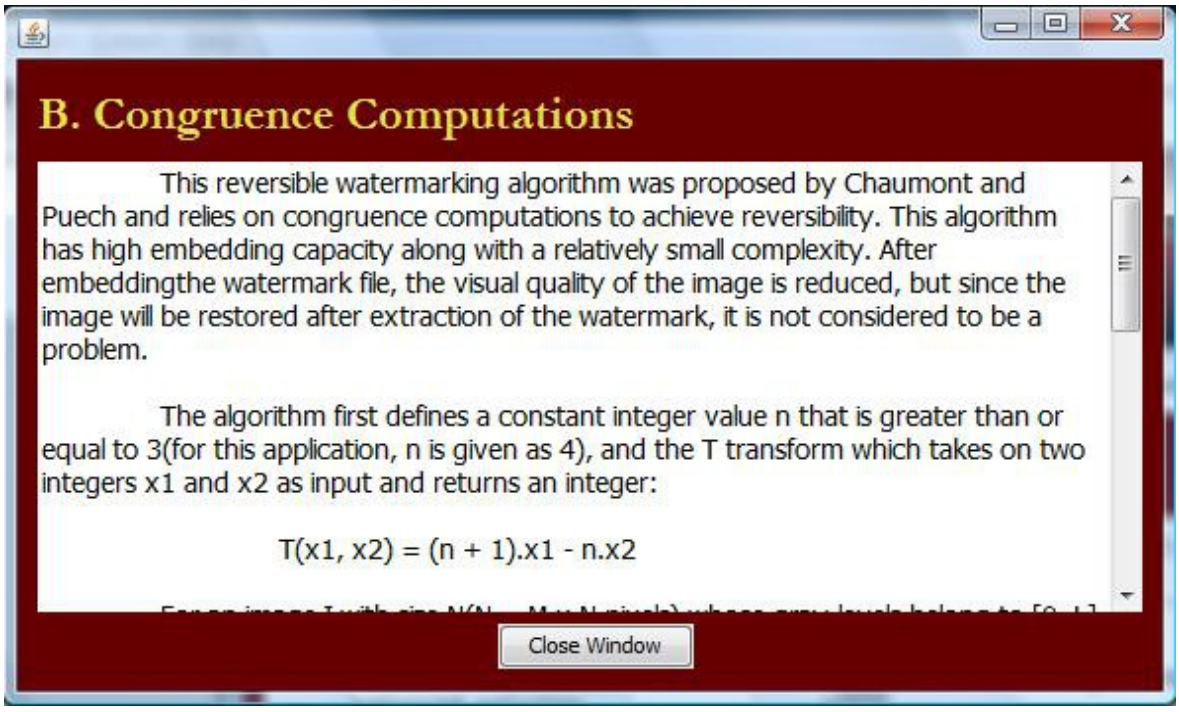


Figure 14: Congruence Computations Help Topic, Reversible Watermarking Application

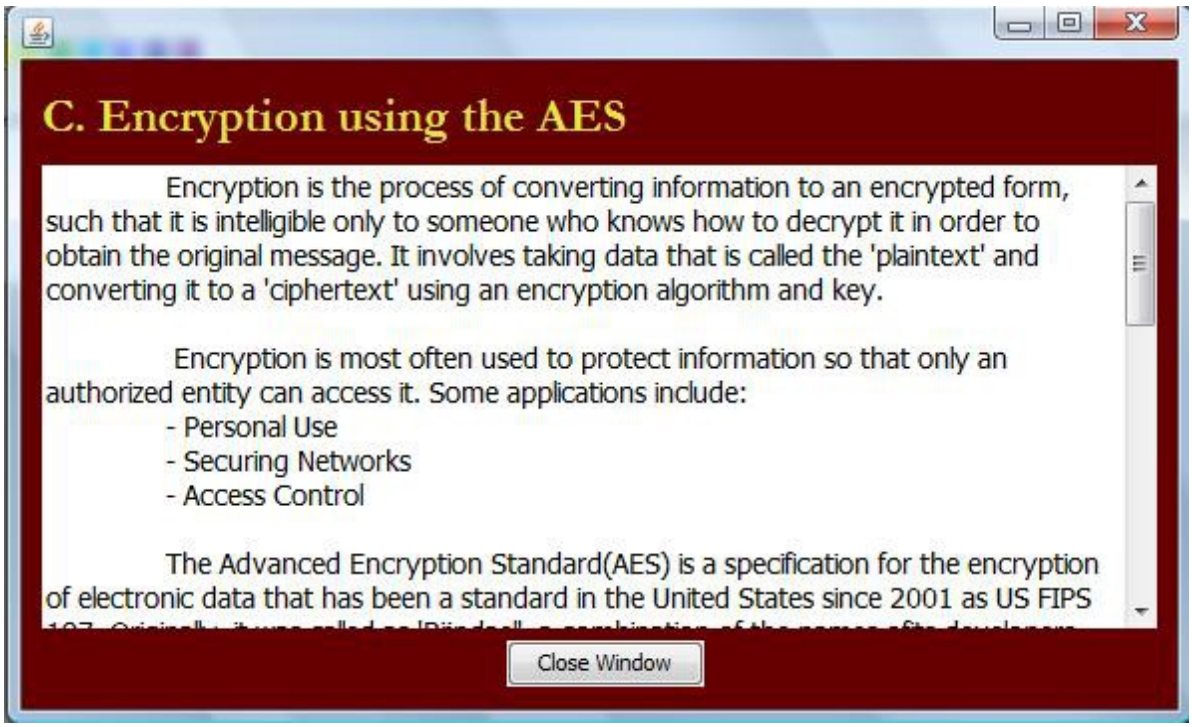


Figure 15: Encryption using the AES Help Topic, Reversible Watermarking Application

The embed watermark page contains a file selector for selecting a single or multiple carrier images, and a file selector for selecting the watermarked file. It also provides an option for the user to change the watermarked image filename, and an option to change the path where the watermarked image will be saved. If the user wishes to encrypt the generated watermarked image, there is an option to do so, and the user needs to input a password that will be used to generate the key. The embed watermark page is illustrated in Figure 16.


Figure 16: Embed Watermark Page, Reversible Watermarking Application

The file selector window for the carrier images, and the file selector window for the watermark file can be seen in Figures 17 and 18 respectively.

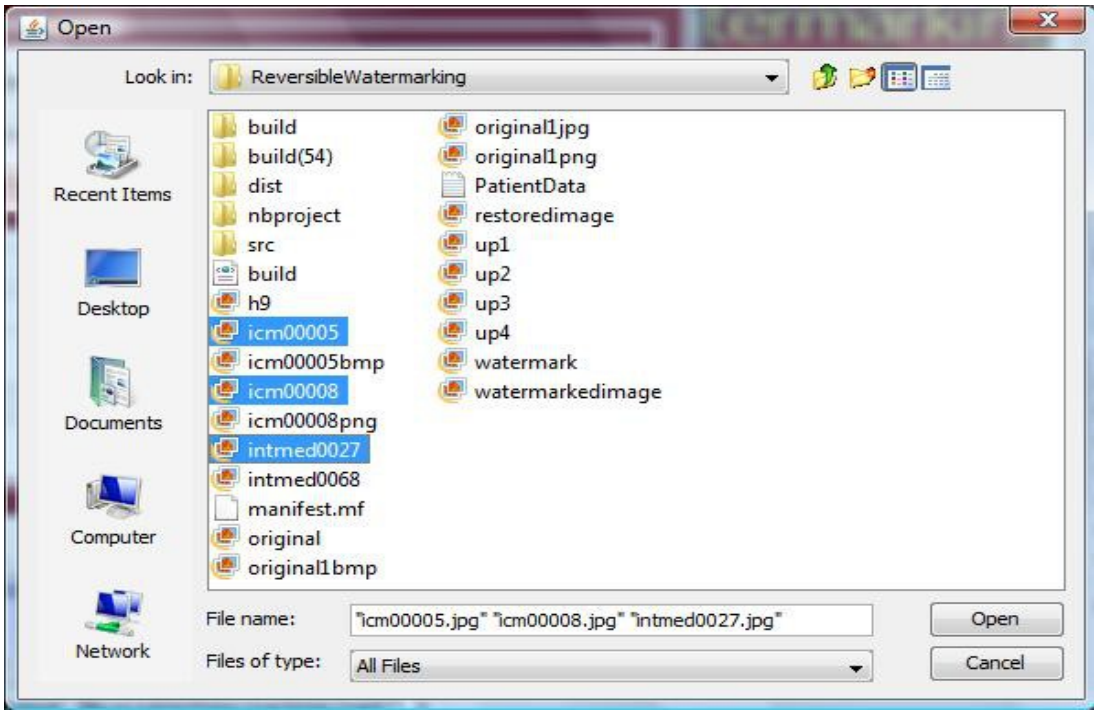


Figure 17: Carrier Image File Selector, Reversible Watermarking Application

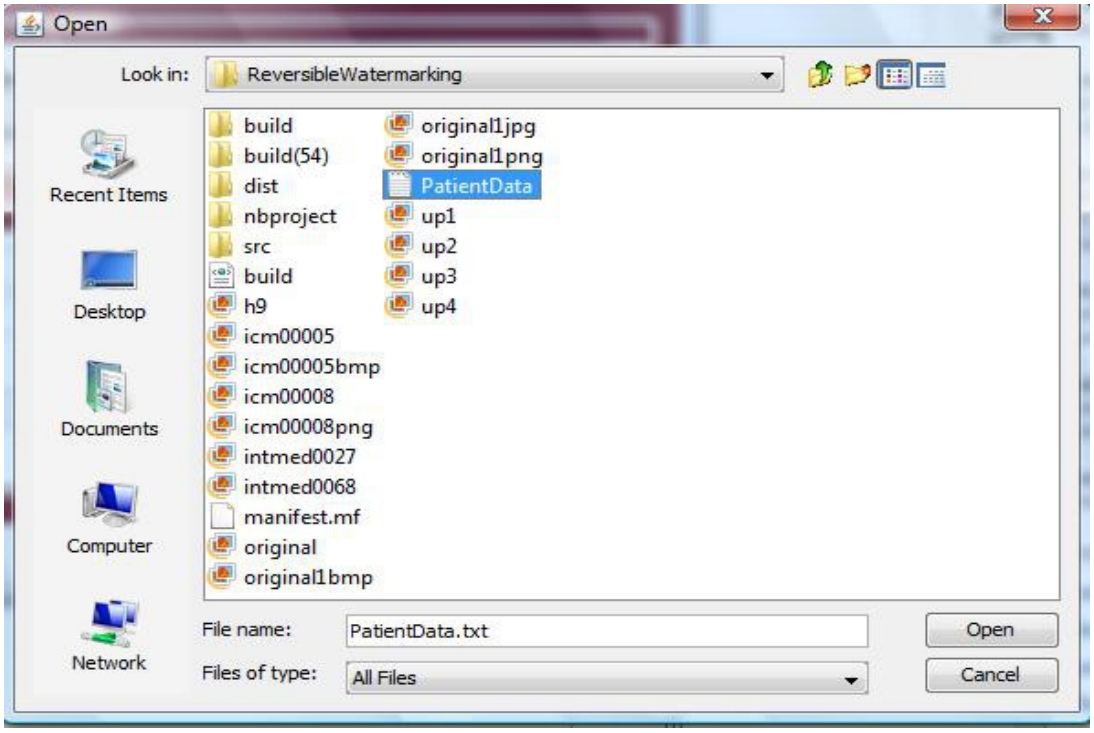


Figure 18: Watermark File Selector, Reversible Watermarking Application

Figures 19 and 20 illustrate the successful results of embedding the watermark. Figure 19 shows a successful watermark embedding without encryption, while Figure 20 shows a successful watermark embedding with encryption.

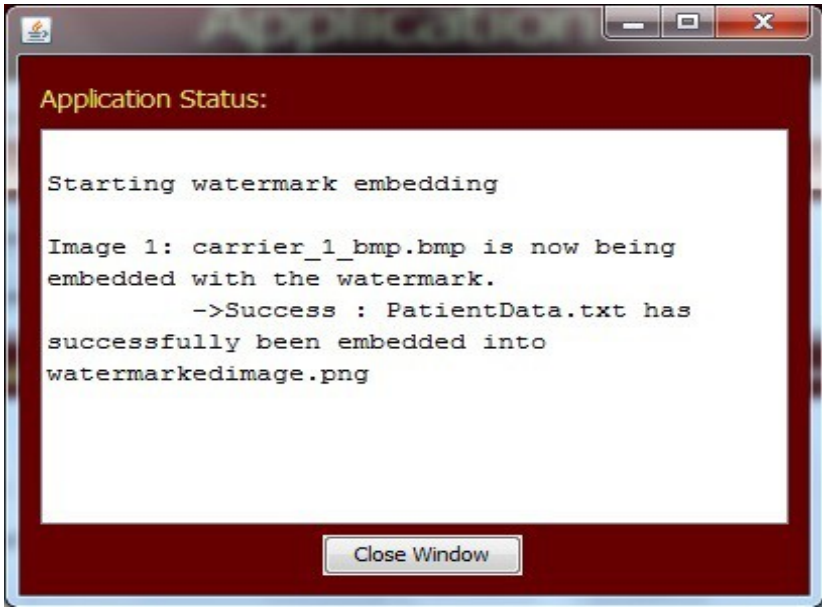


Figure 19: Successful Watermark Embedding without Encryption, Reversible Watermarking Application

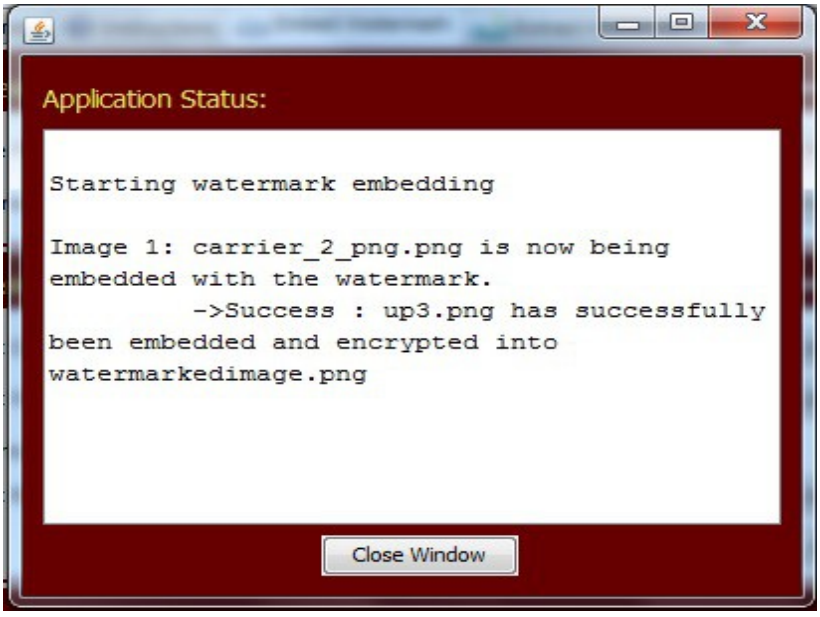


Figure 20: Successful Watermark Embedding with Encryption, Reversible Watermarking Application

The application also shows different status messages if the watermark embedding is not successful. Figure 21 is the status displayed if there is not enough space in the carrier image to embed the watermark.

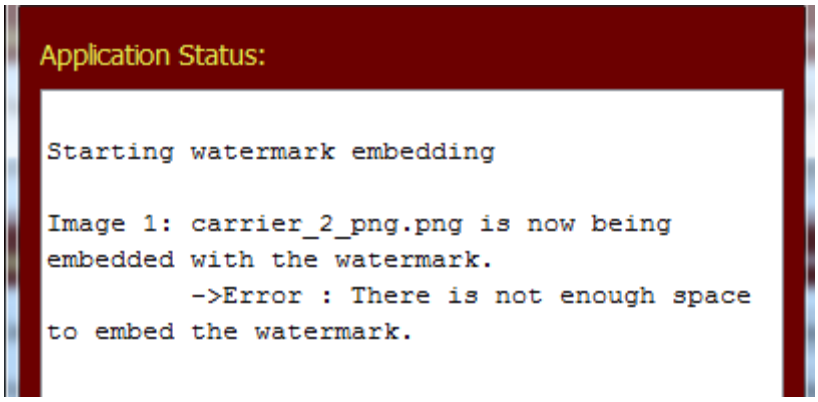


Figure 21: Unsuccessful Watermark Embedding due to Insufficient Space, Reversible Watermarking Application

Other causes of unsuccessful watermark embedding are invalid file type of the carrier image and invalid file type of the watermark. Figures 22 to 25 show the error message for invalid carrier image file type, the error message for invalid watermark file type, and the error message if the carrier image is not grayscale, respectively.

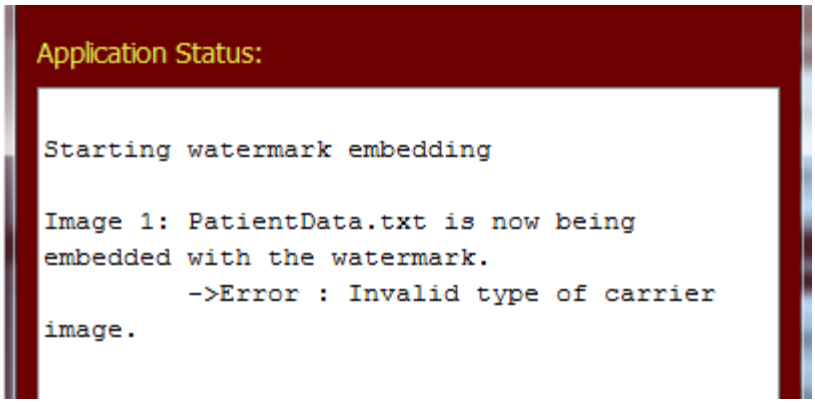


Figure 22: Unsuccessful Watermark Embedding due to Invalid Carrier Image File Type, Reversible Watermarking Application

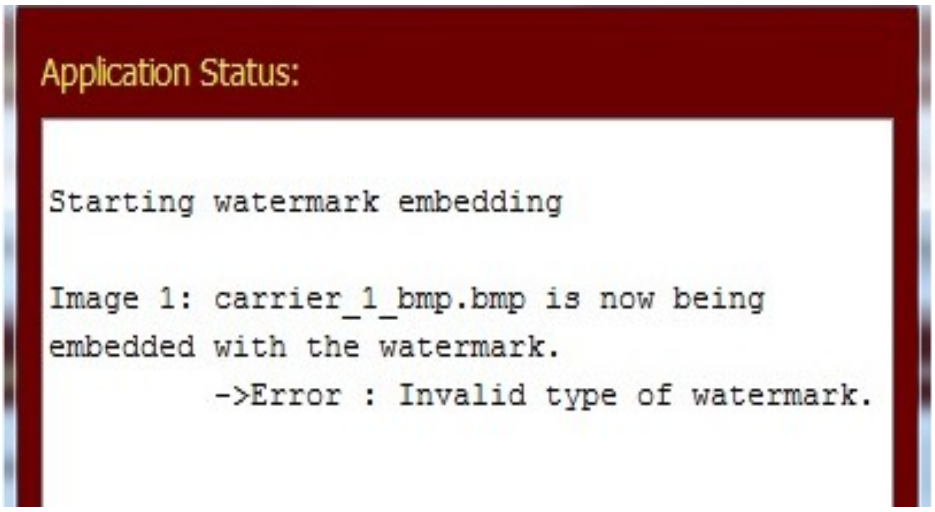


Figure 23: Unsuccessful Watermark Embedding due to Invalid Watermark File Type, Reversible Watermarking Application


Figure 24: Unsuccessful Watermark Embedding due to Invalid Color of Carrier Image, Reversible Watermarking Application

The extract watermark page has a file selector to select a single or multiple watermarked images.It also provides the user the option to change the restored image filename, the extracted watermark filename and the path where they will be saved. If the watermarked image is encrypted, the user can input the password that will be used to generate the key. The extract watermark page can be seen in Figure 25.


Figure 25: Extract Watermark Page, Reversible Watermarking Application

Figures 26 and 27 illustrate the file selector window for the watermarked images and the application status displayed upon successful extraction of the watermark and restoration of the original carrier image, respectively.

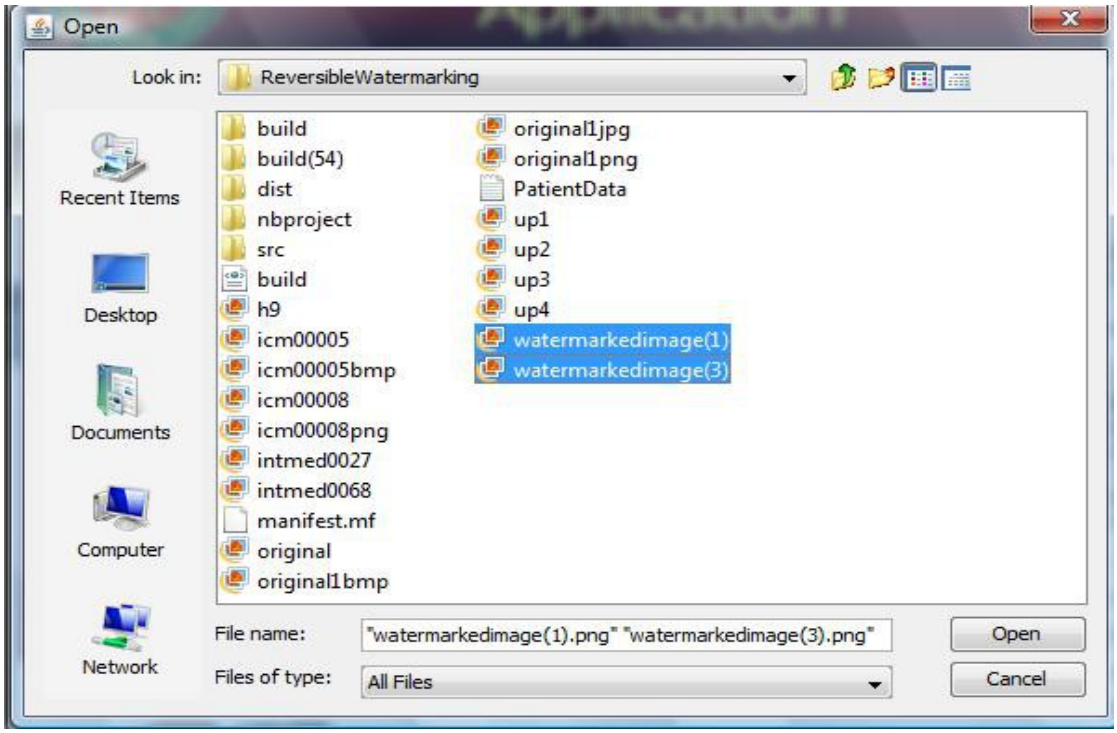


Figure 26: Watermarked Image File Selector, Reversible Watermarking Application

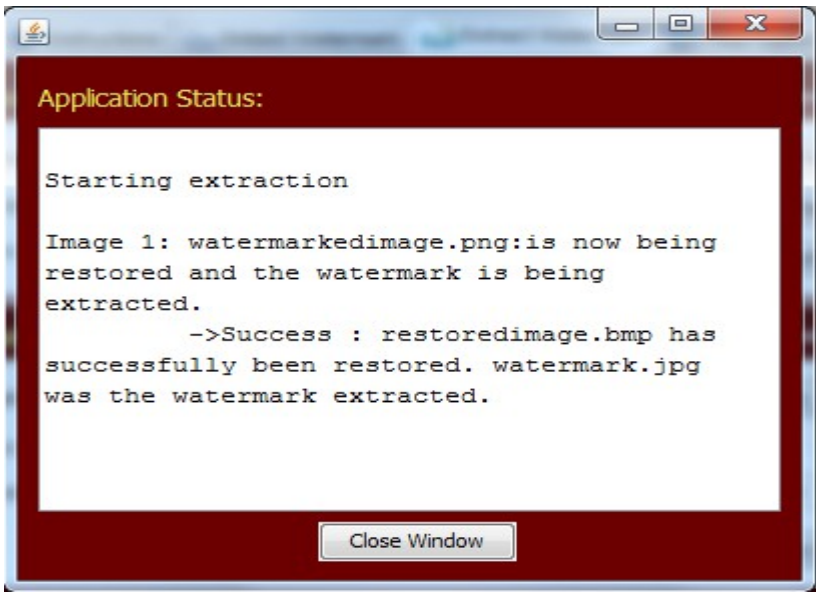


Figure 27: Successful Watermark Extraction, Reversible Watermarking Application

Like the watermark embedding process, the application also displays different statuses if the watermark extraction and image restoration process is not successful. Figures 28 to 30 show the status displayed if the watermark extraction process is unsuccessful due to a wrong password, the status shown if the extraction process is unsuccessful due to an invalid type of watermarked image, and the status shown if the watermark cannot be extracted because the image provided has no watermark or the watermark image has been modified, respectively.

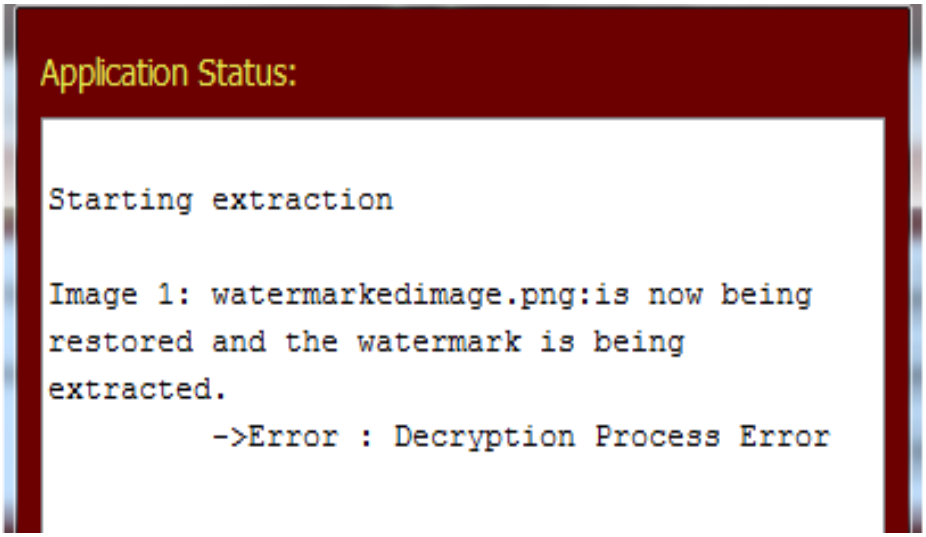


Figure 28: Unsuccessful Watermark Extraction due to Wrong Password, Reversible Watermarking Application


Figure 29: Unsuccessful Watermark Extraction due to Invalid Image File Type,

Reversible Watermarking Application

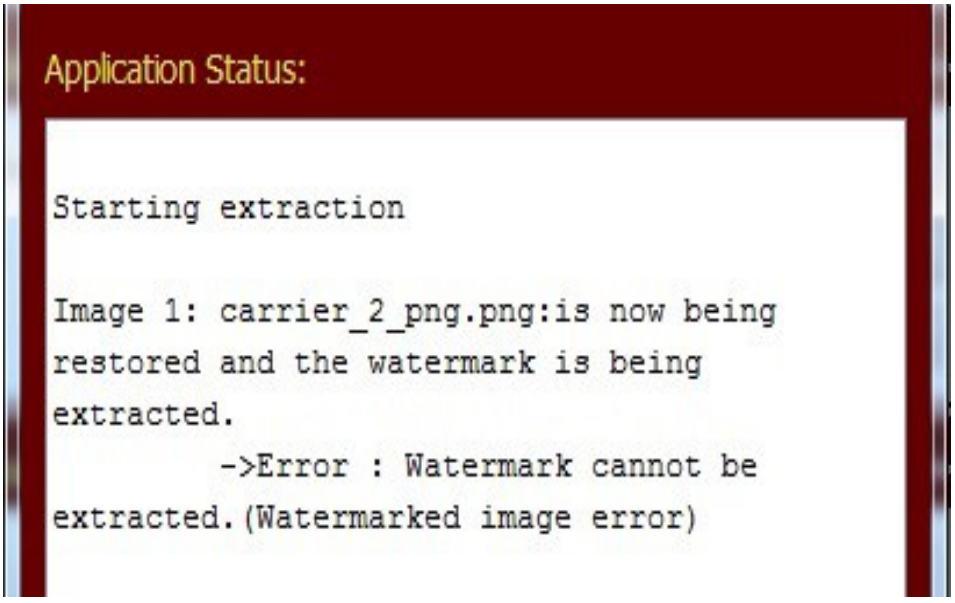


Figure 30: Unsuccessful Watermark Extraction,

Reversible Watermarking Application

# VI. Discussions

The Reversible Watermarking Application for Medical Images allows the user to view instructions, view help topics, embed a watermark file into a carrier image, and extract a watermark and restore a watermarked image. In the instructions page, the application displays how the embed watermark and extract watermark functionalities are used. The view help topics page displays more information about Reversible Watermarking, Congruence Computations, and File Encryption using the Advanced Encryption Standard.

In the embed watermark page, the user can select a single, or multiple images and a watermark file that will be embedded in the selected carrier image(s). If the user wishes to add encryption to the generated watermarked image, the user can provide a password that will then be used to generate the key. The application also provides an option for the user to change the filename of the watermarked image, and to change the folder where it will be saved.

In the extract watermark page, the user selects a single or multiple watermarked images and extracts the embedded file and restores the original carrier image(s). If the watermarked image(s) is encrypted, the user can input the password that will be used to generate the key. The application also allows the user to change the extracted watermark filename, the restored carrier image filename, and the folder where they will be saved.

The application provides a way to embed a watermark file that improves the usage of medical images by improving the security for the images used, making sure that it has not been tampered with. The extracted watermark can serve as an authenticator for the user, and can also carry patient information, making sure that the restored image will be used for the correct person. The Advanced Encryption Standard makes sure that the image is secured if the user chooses to encrypt the watermarked file. The restoration of the original carrier image makes sure that all changes from the watermarking process is removed, and will possibly not cause a wrong diagnosis by the medical personnel in charge. However, for now, the application is limited to carrier images of type JPEG, PNG, and BMP, and watermark files of type TXT, JPEG, PNG, and BMP. It is also only used for grayscale medical images. It is also limited to only a single encryption algorithm, the Advanced Encryption Standard.

Traditional Watermarking methods only approximate the original carrier image. As such, there are permanent distortions to the image which are not acceptable for the medical usage of images. Indeed, previous studies has shown that watermarking has been used for medical images before, however, the use of it has been stalled due to a lack of standard to quantitatively evaluate watermark interference with the diagnosis.[24] Reversible Watermarking methods however, provide a way to restore the original image after watermark extraction, and is considered a solution to the problem of securing medical images.[3] Most algorithms that have been presented are all fragile, meaning that if the image is modified, the watermark is lost. As such, the question of robustness is largely open and it is one of the upcoming challenges for future reversible watermarking algorithms.[25]

The reversible watermarking algorithm of the application, which relies on congruence computations, provides a high embedding capacity for the watermark, as well as a smaller complexity, when compared with other reversible watermarking schemes that has been presented before such as that by Tian[12], Yaqub[14] and Yang[15]. There is also no need to use any compression on the watermark data, unlike the algorithm presented by Liew et. al. which employs Run-Length Encoding, a lossless data compression scheme.[16] The algorithm to be used works with the pixel data of the image, and because of this, even small modifications to the watermarked image might cause the loss of the embedded watermark. As such, the algorithm is said to be fragile. Other algorithms are also implemented using MATLAB, while the application is made using Java, which allows easier set up and usage for the users. The Advanced Encryption Standard also adds a layer of protection to the image, and it is considered very secure as all attacks against it have failed or will take an impossible amount of time succeed.[7]

# VII. Conclusion

The Reversible Watermarking Java Application for medical images is a tool that is designed in order to improve the usage of medical images. It uses Congruence Computations for the watermark embedding and extraction processes. The application allows users to embed a single watermark file into a single or multiple carrier image(s), and select the filename of the generated watermarked file and the location where it will be saved. From a watermarked image, the application can extract the embedded watermark file, and restore the carrier image such that it is exactly the same as the original image. Moreover, the application also provides an optional function which adds security, through encryption of the watermarked image using the Advanced Encryption Standard.

# VIII. Recommendation

The Reversible Watermarking Application allows users to embed a watermark file into a carrier image as well as extract a watermark and restore the original carrier image. The application only allows carrier images of type JPEG, PNG, and BMP. It can be improved by increasing the number of carrier image types it can use, such as GIF and DICOM images. The number of allowed watermark file types can also be increased. It currently accepts watermark files of type TXT, JPEG, PNG, and BMP. Furthermore, the application can be extended to provide support for color images, and not just grayscale, as well as support for dynamic images. The encryption function can also be extended to support different encryption algorithms and not just the AES.

# IX. Bibliography


[1] Zain, J. M. (2005). Digital Watermarking in Medical Images (Doctoral Dissertation). School of Information Systems, Computing and Mathematics, Brunei University.

[2] Umamageswari, A., Ferni Ukrit, M., & Suresh, G.R. (2011). A Survey on Security in Medical Image Communication. International Journal of Computer Applications. 30(3). 41-45.

[3] Narawade, N., & Kanphade, R. (2011). Reversible Watermarking: A Complete Review. International Journal of Computer Science and Telecommunications. 2(3). 46-50.

[4] Chaumont, M., & Puech, W. (2009). A High Capacity Reversible Watermarking Scheme. Electronic Imaging, Visual Communications and Image Processing 2009 SPIE.

[5] United Kingdom Parliament Office of Science and Technology. (2006). Data Encryption. Retrieved October 13, 2012, from http://www.parliament.uk/documents/post/postpn270.pdf

[6] U.S. National Institute of Standards and Technology. (2001). Announcing the ADVANCED ENCRYPTION STANDARD (AES)(Federal Information Processing Standards Publication 197). Retrieved October 9, 2012, from http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[7] Computerworld Inc. (2011). AES Proved Vulnerable by Microsoft Researchers. Retrieved October 9, 2012, from http://www.computerworld.com/s/article/9219297/AES_proved_vulnerable_by_Microsoft_researchers

[8] Portable Network Graphics Home Site. (2012). A Basic Introduction to PNG Features. Retrieved October 13, 2012, from http://www.libpng.org/pub/png/pngintro.html.

[9] Ni, J. (2011). Medical Imaging Informatics. Retrieved from The University of Iowa, Carver College of Medicine Website: http://www.uiowa.edu/hri/courses/medicalImagingInformatics/Spring2011/lecture018.pdf

[10] Feng, J. B., Lin, I. C., Tsai, C. S., & Chu, Y. P. (2006). Reversible Watermarking: Current Status and Key Issues. International Journal of Network Security. 2(3). 161-171.

[11] Caldelli, R., Filipinni, F., & Becarelli, R. (2010) Reversible Watermarking Techniques: An Overview and a Classification. EURASIP Journal on Information Security. 2010.

[12] Tian, J. (2003). Reversible Data Embedding and Content Authentication using Difference Expansion. IEEE Transaction on Circuits and systems for Video Technology, February.

[13] Kallel, I. F., Bouhlel, M. S., & Lapayre, J. C. (2007) Improved Tian's Method for Medical Image Reversible Watermarking. ICGST International Journal on Graphics, Vision and Image Processing. 7(2).

[14] Yaqub, M. K., & Al-jaber, A. (2006). Reversible Watermarking using Modified Difference Expansion. International Journal of Computing and Information Sciences. 4(3). 134-142.

[15] Yang, Y., Sun, X., Yang, H., Li, C., & Xiao, R. (2009) A Contrast-Sensitive Reversible Visible Image Watermarking Technique. IEEE Transactions on Circuits and Systems for Video Technology. 19(5). 656-667.

[16] Liew, S. C., Liew, S. W., & Zain, J. M. (2010). Reversible Medical Image Watermarking For Tamper Detection And Recovery With Run Length Encoding Compression. World Academy of Science, Engineering and Technology 48.

[17] Narawade, N., & Kanphade, R. (2012). DCT Based Robust Reversible Watermarking For Geometric Attack. International Journal of Emerging Trends & Technology in Computer Science. 1(2). 27-32.

[18] Golpira, H., & Danyali, H. (2011). Reversible Medical Image Watermarking based on Wavelet Histogram Shifting. The Imaging Science Journal. 59(1). 49-59.

[19] Cummins, J., Diskin, P., Lau, S., & Parlett, R. (2004). Steganography and Digital Watermarking. Retrieved from The University of Birmingham, School of Computer Science Website: http://www.cs.bham.ac.uk/~mdr/teaching/modules03/security/students/SS5/Steganography.pdf

[20] Fridrich, J., Goljan, M., & Du, R. (2002). Lossless data embedding - new Paradigm in Digital Watermarking. EURASIP Journal on Applied Signal Processing. 2002. 185-196.

[21] Tian, J. (2002). Wavelet-based Reversible Watermarking for Authentication. Security and Watermarking of Multimedia Contents IV. 4675. 679-690.

[22] United States Food and Drug Administration. (2012). Medical Imaging. Retrieved October 1, 2012, from http://www.fda.gov/RadiationEmittingProducts/RadiationEmittingProductsandProcedures/MedicalImaging/default.htm

[23] Medical Imaging and Technology Alliance. (2012). Medical Imaging Primer. Retrieved October 1, 2012, from http://www.medicalimaging.org/medical-imaging-primer/

[24] Coatrieux, G., Lecornu, L., Roux, Ch., & Sankur, B. (2006). A Review of Image Watermarking Applications in Healthcare. 28th Annual International Conference of the IEEE EMBS.

[25] Pan, W., Coatrieux, G., Montagner, J., Cuppens, N., Cuppens, F., & Roux, Ch. (2009). Comparison of Some Reversible Watermarking Methods in Application to Medical Images. 31st Annual International Conference of the IEEE EMBS.

# X. Appendix

## A. Source Codes

```
package my.reversiblewatermarking;

import java.awt.Component;
import java.awt.Graphics;
import java.awt.Image;
import java.io.File;
import javax.swing.ImageIcon;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.SwingWorker;

/**
 *
 * @author Dan
 */
public class ReversibleWatermarkingUI extends
javax.swing.JFrame {
    private Component parentFrame;

    /**
     * Creates new form ReversibleWatermarkingUI
     */
    public ReversibleWatermarkingUI() {
        master = new Application_handler();
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this
method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        encryptionkey_browseButton = new javax.swing.JButton();
        decryptionkey_browseButton = new javax.swing.JButton();
        titlePanel = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        applicationPanel = new javax.swing.JPanel();
        applicationTabbedPane = new javax.swing.JTabbedPane();
        welcomeTab = new javax.swing.JPanel();
        welcomeTab_InnerPanel = new BackgroundPanel();
        welcomedesc_Label = new javax.swing.JLabel();
        instructionsdesc_Label = new javax.swing.JLabel();
        embeddesc_Label = new javax.swing.JLabel();
        helpdesc_Label = new javax.swing.JLabel();
        extractdesc_Label = new javax.swing.JLabel();
        wlecometext_Label = new javax.swing.JLabel();
        instructTab = new javax.swing.JPanel();
        embed_instructionsPanel = new javax.swing.JPanel();
        embed_inst1 = new javax.swing.JLabel();
        embed_inst2 = new javax.swing.JLabel();
        embed_inst3 = new javax.swing.JLabel();
        embed_inst4 = new javax.swing.JLabel();
        embed_inst5 = new javax.swing.JLabel();
        embed_inst6 = new javax.swing.JLabel();
        extract_instructionsPanel = new javax.swing.JPanel();
        extract_inst1 = new javax.swing.JLabel();
        extract_inst2 = new javax.swing.JLabel();
        extract_inst3 = new javax.swing.JLabel();
        extract_inst4 = new javax.swing.JLabel();
        extract_inst5 = new javax.swing.JLabel();
        embedTab = new javax.swing.JPanel();
        embed_settingsPanel = new javax.swing.JPanel();
        embed_settingInnerPanel = new javax.swing.JPanel();
        def_outimageButton = new javax.swing.JButton();
        outpathTextArea = new javax.swing.JTextField();
        outimage_setButton = new javax.swing.JButton();
        outpath_defButton = new javax.swing.JButton();
        outpath_browseButton = new javax.swing.JButton();
        outimageTextArea = new javax.swing.JTextField();
        outimagenameLabel = new javax.swing.JLabel();
        outpathLabel = new javax.swing.JLabel();
        embed_imagesPanel = new javax.swing.JPanel();
        embed_imagesInnerPanel = new javax.swing.JPanel();
        embed_clearButton = new javax.swing.JButton();
        embed_startButton = new javax.swing.JButton();
        encryptionkey_setButton = new javax.swing.JButton();
        encryptionkey_clearButton = new javax.swing.JButton();
        encryptionkeyTextArea = new javax.swing.JTextField();
        encryptionkeyLabel = new javax.swing.JLabel();
        encryptionkeyCheckBox = new javax.swing.JCheckBox();
        watermarkLabel = new javax.swing.JLabel();
        watermarkTextArea = new javax.swing.JTextField();
        watermark_browseButton = new javax.swing.JButton();
        watermark_clearButton = new javax.swing.JButton();
        carrierimage_clearButton = new javax.swing.JButton();
        carrierimage_browseButton = new javax.swing.JButton();
        carrierimageLabel = new javax.swing.JLabel();
        carrierimageTextArea = new javax.swing.JTextField();
        extractTab = new javax.swing.JPanel();
        extract_settingsPanel = new javax.swing.JPanel();
        extract_settingsInnerPanel = new javax.swing.JPanel();
        restoredwatermark_setButton = new javax.swing.JButton();
        restoredimageLabel = new javax.swing.JLabel();
        restoredpathTextArea = new javax.swing.JTextField();
        restoredwatermarkLabel = new javax.swing.JLabel();
        restoredimageTextArea = new javax.swing.JTextField();
        def_restoredwatermarkButton = new javax.swing.JButton();
        def_restoredimageButton = new javax.swing.JButton();
        restoredwatermarkTextArea = new javax.swing.JTextField();
        restoredpath_defButton = new javax.swing.JButton();
        restoredpathLabel = new javax.swing.JLabel();
        restoredimage_setButton = new javax.swing.JButton();
        restoredpath_browseButton = new javax.swing.JButton();
        extract_imagesPanel = new javax.swing.JPanel();
        extract_imagesInnerPanel = new javax.swing.JPanel();
        inputimageLabel = new javax.swing.JLabel();
        decryptionkeyLabel = new javax.swing.JLabel();
        decryptionkey_clearButton = new javax.swing.JButton();
        decryptionkeyTextArea = new javax.swing.JTextField();
        inputimage_clearButton = new javax.swing.JButton();
        inputimage_browseButton = new javax.swing.JButton();
        decryptionkeyCheckBox = new javax.swing.JCheckBox();
```

```
        inputimageTextArea = new javax.swing.JTextField();
        extract_startButton = new javax.swing.JButton();
        extract_clearButton = new javax.swing.JButton();
        decryptionkey_setButton = new javax.swing.JButton();
        helpTab = new javax.swing.JPanel();
        helptext_Label = new javax.swing.JLabel();
        help_InnerPanel = new BackgroundPanel();
        helpinst_Label = new javax.swing.JLabel();
        helpA_Label = new javax.swing.JLabel();
        helpB_Label = new javax.swing.JLabel();
        helpC_Label = new javax.swing.JLabel();
        exitapplicationButton = new javax.swing.JButton();

        encryptionkey_browseButton.setText("Browse Files");
        encryptionkey_browseButton.setEnabled(false);
        encryptionkey_browseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                encryptionkey_browseButtonActionPerformed(evt);
            }
        });

        decryptionkey_browseButton.setText("Browse Files");
        decryptionkey_browseButton.setEnabled(false);
        decryptionkey_browseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                decryptionkey_browseButtonActionPerformed(evt);
            }
        });


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_
ON_CLOSE);
        setResizable(false);

        titlePanel.setMaximumSize(new java.awt.Dimension(579,
102));
        titlePanel.setMinimumSize(new java.awt.Dimension(579,
102));

        jLabel1.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/header.png"))); // NOI18N

        javax.swing.GroupLayout titlePanelLayout = new
javax.swing.GroupLayout(titlePanel);
        titlePanel.setLayout(titlePanelLayout);
        titlePanelLayout.setHorizontalGroup(

titlePanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        titlePanelLayout.setVerticalGroup(

titlePanelLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        welcomeTab.setBackground(new java.awt.Color(102, 0, 0));

        welcomeTab_InnerPanel.setBackground(new
java.awt.Color(255, 255, 255));

        welcomedesc_Label.setFont(new java.awt.Font("Tahoma",
0, 14)); // NOI18N
        welcomedesc_Label.setText("<html>This application allows
you to invisibly embed a watermark to medical images and
restore them perfectly upon extracting the
watermark.<br><hr></html>");

        instructionsdesc_Label.setFont(new
java.awt.Font("Tahoma", 0, 12)); // NOI18N
        instructionsdesc_Label.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/instrbig.png"))); // NOI18N
        instructionsdesc_Label.setText("<html><b>View
Instructions</b><br>The instructions page provides good
information for first time users of the Reversible Watermarking
Application.</html>");
        instructionsdesc_Label.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
                instructionsdesc_LabelMouseClicked(evt);
            }
        });

        embeddesc_Label.setFont(new java.awt.Font("Tahoma", 0,
12)); // NOI18N
        embeddesc_Label.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/embedbig.gif"))); // NOI18N
        embeddesc_Label.setText("<html><b>Embed
Watermark</b><br>Start embedding watermark files in carrier
images.</html>");

embeddesc_Label.setVerticalAlignment(javax.swing.SwingConst
ants.TOP);
        embeddesc_Label.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
                embeddesc_LabelMouseClicked(evt);
            }
        });

        helpdesc_Label.setFont(new java.awt.Font("Tahoma", 0,
12)); // NOI18N
        helpdesc_Label.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/helpbig.jpg"))); // NOI18N
        helpdesc_Label.setText("<html><b>View
 Help Topics</b><br>Read more about reversible watermarking,
congruence computations, and other information about the
application.</html>");
        helpdesc_Label.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
                helpdesc_LabelMouseClicked(evt);
            }
        });

        extractdesc_Label.setFont(new java.awt.Font("Tahoma", 0,
12)); // NOI18N
        extractdesc_Label.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/extractbig.gif"))); // NOI18N
        extractdesc_Label.setText("<html><b>Extract
Watermark</b><br>Extract an embedded file in a watermarked
image and restore its original pixel data.</html>");
```

```
extractdesc_Label.setVerticalAlignment(javax.swing.SwingConst
ants.TOP);
        extractdesc_Label.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
                extractdesc_LabelMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout welcomeTab_InnerPanelLayout =
new javax.swing.GroupLayout(welcomeTab_InnerPanel);

welcomeTab_InnerPanel.setLayout(welcomeTab_InnerPanelLay
out);
        welcomeTab_InnerPanelLayout.setHorizontalGroup(

welcomeTab_InnerPanelLayout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(welcomeTab_InnerPanelLayout.createSequentialGro
up()
                .addContainerGap()

.addGroup(welcomeTab_InnerPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(welcomedesc_Label,
javax.swing.GroupLayout.DEFAULT_SIZE, 539,
Short.MAX_VALUE)

.addGroup(welcomeTab_InnerPanelLayout.createSequentialGro
up()

.addGroup(welcomeTab_InnerPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(helpdesc_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)
                        .addComponent(instructionsdesc_Label,
javax.swing.GroupLayout.DEFAULT_SIZE, 267,
Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(welcomeTab_InnerPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(embeddesc_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)
                        .addComponent(extractdesc_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE))))
                .addContainerGap())
        );
        welcomeTab_InnerPanelLayout.setVerticalGroup(

welcomeTab_InnerPanelLayout.createParallelGroup(javax.swing
.GroupLayout.Alignment.LEADING)

.addGroup(welcomeTab_InnerPanelLayout.createSequentialGro
up()
                .addContainerGap()
                .addComponent(welcomedesc_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 53,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(welcomeTab_InnerPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(instructionsdesc_Label,
javax.swing.GroupLayout.DEFAULT_SIZE, 87,
Short.MAX_VALUE)
                    .addComponent(embeddesc_Label))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, 9, Short.MAX_VALUE)

.addGroup(welcomeTab_InnerPanelLayout.createParallelGroup(
javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(helpdesc_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 87,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(extractdesc_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 78,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );

        wlecometext_Label.setFont(new
java.awt.Font("Garamond", 1, 24)); // NOI18N
        wlecometext_Label.setForeground(new java.awt.Color(230,
230, 71));
        wlecometext_Label.setText("Welcome!");

        javax.swing.GroupLayout welcomeTabLayout = new
javax.swing.GroupLayout(welcomeTab);
        welcomeTab.setLayout(welcomeTabLayout);
        welcomeTabLayout.setHorizontalGroup(

welcomeTabLayout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)
            .addGroup(welcomeTabLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(welcomeTabLayout.createParallelGroup(javax.swing.
GroupLayout.Alignment.LEADING)
                    .addComponent(welcomeTab_InnerPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(wlecometext_Label,
javax.swing.GroupLayout.DEFAULT_SIZE, 559,
Short.MAX_VALUE))
                .addContainerGap())
        );
        welcomeTabLayout.setVerticalGroup(

welcomeTabLayout.createParallelGroup(javax.swing.GroupLayo
ut.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
welcomeTabLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(wlecometext_Label,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(welcomeTab_InnerPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(156, 156, 156))
        );

        applicationTabbedPane.addTab("Welcome!", new
```

```
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/Home_icon.png")), welcomeTab); // NOI18N

        instructTab.setBackground(new java.awt.Color(102, 0, 0));

        embed_instructionsPanel.setBackground(new
java.awt.Color(102, 0, 0));

embed_instructionsPanel.setBorder(javax.swing.BorderFactory.c
reateTitledBorder(new javax.swing.border.LineBorder(new
java.awt.Color(255, 255, 255), 3, true), "Watermark Embedding
Instructions",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 0, 14), new java.awt.Color(230, 230,
71))); // NOI18N

        embed_inst1.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        embed_inst1.setForeground(new java.awt.Color(222, 222,
222));
        embed_inst1.setText("To start the watermark embedding
process:");

        embed_inst2.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        embed_inst2.setForeground(new java.awt.Color(222, 222,
222));
        embed_inst2.setText("1. Select the carrier file and the
watermark file using the 'Browse Files' button.");

        embed_inst3.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        embed_inst3.setForeground(new java.awt.Color(222, 222,
222));
        embed_inst3.setText("2. Provide a password that will be
used to generate the encryption key.");

        embed_inst4.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        embed_inst4.setForeground(new java.awt.Color(222, 222,
222));
        embed_inst4.setText("OPTIONAL:");

        embed_inst5.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        embed_inst5.setForeground(new java.awt.Color(222, 222,
222));
        embed_inst5.setText("3. Provide the filename and filepath
for the watermarked image.");

        embed_inst6.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        embed_inst6.setForeground(new java.awt.Color(222, 222,
222));
        embed_inst6.setText("4. Press the \"Embed Watermark\"
button.");

        javax.swing.GroupLayout embed_instructionsPanelLayout
= new javax.swing.GroupLayout(embed_instructionsPanel);

embed_instructionsPanel.setLayout(embed_instructionsPanelLa
yout);
        embed_instructionsPanelLayout.setHorizontalGroup(

embed_instructionsPanelLayout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(embed_instructionsPanelLayout.createSequentialGr
oup()
            .addContainerGap()

.addGroup(embed_instructionsPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(embed_instructionsPanelLayout.createSequentialGr
oup()
                .addGap(10, 10, 10)

.addGroup(embed_instructionsPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(embed_inst2,
javax.swing.GroupLayout.DEFAULT_SIZE, 515,
Short.MAX_VALUE)
                    .addComponent(embed_inst6)

.addGroup(embed_instructionsPanelLayout.createSequentialGr
oup()
                        .addComponent(embed_inst4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(embed_instructionsPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(embed_inst5)
                            .addComponent(embed_inst3)))))

.addGroup(embed_instructionsPanelLayout.createSequentialGr
oup()
                .addComponent(embed_inst1)
                .addGap(0, 0, Short.MAX_VALUE)))
            .addContainerGap())
        );
        embed_instructionsPanelLayout.setVerticalGroup(

embed_instructionsPanelLayout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(embed_instructionsPanelLayout.createSequentialGr
oup()
            .addComponent(embed_inst1)


.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
            .addComponent(embed_inst2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(embed_instructionsPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(embed_inst4)
                .addComponent(embed_inst3,
javax.swing.GroupLayout.PREFERRED_SIZE, 17,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
            .addComponent(embed_inst5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
            .addComponent(embed_inst6)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
```

```
        extract_instructionsPanel.setBackground(new
java.awt.Color(102, 0, 0));

extract_instructionsPanel.setBorder(javax.swing.BorderFactory.c
reateTitledBorder(new javax.swing.border.LineBorder(new
java.awt.Color(255, 255, 255), 3, true), "Watermark Extraction
Instructions",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 0, 14), new java.awt.Color(230, 230,
71))); // NOI18N

        extract_inst1.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        extract_inst1.setForeground(new java.awt.Color(222, 222,
222));
        extract_inst1.setText("To start watermark extraction and
image restoration:");

        extract_inst2.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        extract_inst2.setForeground(new java.awt.Color(222, 222,
222));
        extract_inst2.setText("1. Select the watermarked image
using the 'Browse Files' button.");

        extract_inst3.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        extract_inst3.setForeground(new java.awt.Color(222, 222,
222));
        extract_inst3.setText("OPTIONAL:  2. If the image is to be
decrypted first, provide the password used.");

        extract_inst4.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        extract_inst4.setForeground(new java.awt.Color(222, 222,
222));
        extract_inst4.setText("3. Provide the filename and path for
the image and the watermark.");

        extract_inst5.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        extract_inst5.setForeground(new java.awt.Color(222, 222,
222));
        extract_inst5.setText("4. Press the 'Extract Watermark'
button.");

        javax.swing.GroupLayout extract_instructionsPanelLayout
= new javax.swing.GroupLayout(extract_instructionsPanel);

extract_instructionsPanel.setLayout(extract_instructionsPanelLa
yout);
        extract_instructionsPanelLayout.setHorizontalGroup(

extract_instructionsPanelLayout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(extract_instructionsPanelLayout.createSequentialGro
up()

.addGroup(extract_instructionsPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(extract_instructionsPanelLayout.createSequentialGro
up()
                        .addContainerGap()

.addGroup(extract_instructionsPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(extract_inst1,
javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
extract_instructionsPanelLayout.createSequentialGroup()
                                .addGap(10, 10, 10)

.addGroup(extract_instructionsPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(extract_inst3)
                                    .addComponent(extract_inst2)
                                    .addComponent(extract_inst5,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))))

.addGroup(extract_instructionsPanelLayout.createSequentialGro
up()
                        .addGap(96, 96, 96)
                        .addComponent(extract_inst4,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
                    .addContainerGap())
        );
        extract_instructionsPanelLayout.setVerticalGroup(

extract_instructionsPanelLayout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(extract_instructionsPanelLayout.createSequentialGro
up()
                    .addComponent(extract_inst1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                    .addComponent(extract_inst2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                    .addComponent(extract_inst3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                    .addComponent(extract_inst4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(extract_inst5)
                    .addContainerGap())
        );

        javax.swing.GroupLayout instructTabLayout = new
javax.swing.GroupLayout(instructTab);
        instructTab.setLayout(instructTabLayout);
        instructTabLayout.setHorizontalGroup(

instructTabLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)
                .addGroup(instructTabLayout.createSequentialGroup()
                    .addContainerGap()

.addGroup(instructTabLayout.createParallelGroup(javax.swing.G
roupLayout.Alignment.LEADING)
                        .addComponent(extract_instructionsPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(embed_instructionsPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
```

```
            .addContainerGap())
        );
        instructTabLayout.setVerticalGroup(

instructTabLayout.createParallelGroup(javax.swing.GroupLayout
.Alignment.LEADING)
            .addGroup(instructTabLayout.createSequentialGroup()
                .addGap(5, 5, 5)
                .addComponent(embed_instructionsPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(extract_instructionsPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );

        applicationTabbedPane.addTab("Instructions", new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/instructions.jpg")), instructTab); // NOI18N

        embedTab.setBackground(new java.awt.Color(102, 0, 0));

        embed_settingsPanel.setBackground(new
java.awt.Color(102, 0, 0));

embed_settingsPanel.setBorder(javax.swing.BorderFactory.crea
teTitledBorder(new javax.swing.border.LineBorder(new
java.awt.Color(255, 255, 255), 3, true), "Embedding Settings",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 14), new java.awt.Color(230, 230,
71))); // NOI18N

        embed_settingInnerPanel.setBackground(new
java.awt.Color(255, 255, 255));

        def_outimageButton.setText("Use Default Filename");
        def_outimageButton.setOpaque(false);
        def_outimageButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                def_outimageButtonActionPerformed(evt);
            }
        });

        outpathTextArea.setEditable(false);
        outpathTextArea.setBackground(new java.awt.Color(255,
255, 255));
        outpathTextArea.setText(master.embedder.get_filepath());
        outpathTextArea.setCursor(new
java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        outpathTextArea.setMaximumSize(new
java.awt.Dimension(6, 20));
        outpathTextArea.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                outpathTextAreaMouseEntered(evt);
            }
        });
        outpathTextArea.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                outpathTextAreaActionPerformed(evt);
            }
        });

        outimage_setButton.setText("Set");
        outimage_setButton.setOpaque(false);
        outimage_setButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                outimage_setButtonActionPerformed(evt);
            }
        });

        outpath_defButton.setText("Use Default Path");
        outpath_defButton.setOpaque(false);
        outpath_defButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                outpath_defButtonActionPerformed(evt);
            }
        });

        outpath_browseButton.setText("Browse");
        outpath_browseButton.setOpaque(false);
        outpath_browseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                outpath_browseButtonActionPerformed(evt);
            }
        });


outimageTextArea.setText(master.embedder.get_filename());
        outimageTextArea.addMouseListener(new
java.awt.event.MouseAdapter()
 {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                outimageTextAreaMouseEntered(evt);
            }
        });
        outimageTextArea.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                outimageTextAreaActionPerformed(evt);
            }
        });

        outimagenameLabel.setText("Image Filename");

        outpathLabel.setText("File Directory");

        javax.swing.GroupLayout embed_settingInnerPanelLayout
= new javax.swing.GroupLayout(embed_settingInnerPanel);

embed_settingInnerPanel.setLayout(embed_settingInnerPanelL
ayout);
        embed_settingInnerPanelLayout.setHorizontalGroup(

embed_settingInnerPanelLayout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
embed_settingInnerPanelLayout.createSequentialGroup()
                .addContainerGap()
```

```
.addGroup(embed_settingInnerPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(outimagenameLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(outpathLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.UNRELATED)

.addGroup(embed_settingInnerPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(outpathTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 171,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(outimageTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 171,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.UNRELATED)

.addGroup(embed_settingInnerPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(outpath_browseButton)
                .addComponent(outimage_setButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(embed_settingInnerPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(outpath_defButton)
                .addComponent(def_outimageButton))
            .addGap(49, 49, 49))
        );
        embed_settingInnerPanelLayout.setVerticalGroup(

embed_settingInnerPanelLayout.createParallelGroup(javax.swin
g.GroupLayout.Alignment.LEADING)

.addGroup(embed_settingInnerPanelLayout.createSequentialGr
oup()
            .addContainerGap()

.addGroup(embed_settingInnerPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(embed_settingInnerPanelLayout.createSequentialGr
oup()
                .addComponent(outimage_setButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(outpath_browseButton))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
embed_settingInnerPanelLayout.createSequentialGroup()
                .addComponent(outimageTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(outpathTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
embed_settingInnerPanelLayout.createSequentialGroup()

.addGroup(embed_settingInnerPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(def_outimageButton)
                .addComponent(outimagenameLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(embed_settingInnerPanelLayout.createParallelGroup
(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(embed_settingInnerPanelLayout.createSequentialGr
oup()
                    .addGap(0, 1, Short.MAX_VALUE)
                    .addComponent(outpath_defButton))
                .addComponent(outpathLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
            .addContainerGap())
        );

        javax.swing.GroupLayout embed_settingsPanelLayout =
new javax.swing.GroupLayout(embed_settingsPanel);

embed_settingsPanel.setLayout(embed_settingsPanelLayout);
        embed_settingsPanelLayout.setHorizontalGroup(

embed_settingsPanelLayout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.LEADING)
            .addComponent(embed_settingInnerPanel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        embed_settingsPanelLayout.setVerticalGroup(

embed_settingsPanelLayout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.LEADING)
            .addComponent(embed_settingInnerPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        embed_imagesPanel.setBackground(new
java.awt.Color(102, 0, 0));

embed_imagesPanel.setBorder(javax.swing.BorderFactory.creat
eTitledBorder(new javax.swing.border.LineBorder(new
java.awt.Color(255, 255, 255), 3, true), "Watermark Embedding",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 14), new java.awt.Color(230, 230,
71))); // NOI18N

        embed_imagesInnerPanel.setBackground(new
java.awt.Color(255, 255, 255));

        embed_clearButton.setText("Clear Selections");
        embed_clearButton.setOpaque(false);
        embed_clearButton.addActionListener(new
```

```
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                embed_clearButtonActionPerformed(evt);
            }
        });

        embed_startButton.setText("Embed Watermark");
        embed_startButton.setOpaque(false);
        embed_startButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                embed_startButtonActionPerformed(evt);
            }
        });

        encryptionkey_setButton.setText("Set Key");
        encryptionkey_setButton.setEnabled(false);
        encryptionkey_setButton.setOpaque(false);
        encryptionkey_setButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                encryptionkey_setButtonActionPerformed(evt);
            }
        });

        encryptionkey_clearButton.setText("Clear");
        encryptionkey_clearButton.setEnabled(false);
        encryptionkey_clearButton.setOpaque(false);
        encryptionkey_clearButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                encryptionkey_clearButtonActionPerformed(evt);
            }
        });

        encryptionkeyTextArea.setEditable(false);
        encryptionkeyTextArea.setMaximumSize(new
java.awt.Dimension(6, 20));
        encryptionkeyTextArea.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                encryptionkeyTextAreaMouseEntered(evt);
            }
        });

        encryptionkeyLabel.setText("Select Encryption Key:");

        encryptionkeyCheckBox.setText("Encrypt Watermarked
Image?");
        encryptionkeyCheckBox.setOpaque(false);
        encryptionkeyCheckBox.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                encryptionkeyCheckBoxActionPerformed(evt);
            }
        });

        watermarkLabel.setText("Select Watermark Image:");

        watermarkTextArea.setEditable(false);
        watermarkTextArea.setBackground(new
java.awt.Color(255, 255, 255));
        watermarkTextArea.setMaximumSize(new
java.awt.Dimension(6, 20));
        watermarkTextArea.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                watermarkTextAreaMouseEntered(evt);
            }
        });

        watermark_browseButton.setText("Browse Files");
        watermark_browseButton.setOpaque(false);
        watermark_browseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                watermark_browseButtonActionPerformed(evt);
            }
        });

        watermark_clearButton.setText("Clear");
        watermark_clearButton.setOpaque(false);
        watermark_clearButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                watermark_clearButtonActionPerformed(evt);
            }
        });

        carrierimage_clearButton.setText("Clear");

        carrierimage_clearButton.setOpaque(false);
        carrierimage_clearButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                carrierimage_clearButtonActionPerformed(evt);
            }
        });

        carrierimage_browseButton.setText("Browse Files");
        carrierimage_browseButton.setOpaque(false);
        carrierimage_browseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                carrierimage_browseButtonActionPerformed(evt);
            }
        });

        carrierimageLabel.setText("Select Carrier Image(s):");

        carrierimageTextArea.setEditable(false);
        carrierimageTextArea.setBackground(new
java.awt.Color(255, 255, 255));
        carrierimageTextArea.setMaximumSize(new
java.awt.Dimension(6, 20));
        carrierimageTextArea.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                carrierimageTextAreaMouseEntered(evt);
            }
        });

        javax.swing.GroupLayout embed_imagesInnerPanelLayout
= new javax.swing.GroupLayout(embed_imagesInnerPanel);

embed_imagesInnerPanel.setLayout(embed_imagesInnerPanel
Layout);
        embed_imagesInnerPanelLayout.setHorizontalGroup(
```

```
embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(embed_imagesInnerPanelLayout.createSequentialGroup()
            .addContainerGap()

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(embed_imagesInnerPanelLayout.createSequentialGroup()
                    .addGap(0, 289, Short.MAX_VALUE)
                    .addComponent(embed_startButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(embed_clearButton))

.addGroup(embed_imagesInnerPanelLayout.createSequentialGroup()

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(encryptionkeyCheckBox)

.addGroup(embed_imagesInnerPanelLayout.createSequentialGroup()

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(embed_imagesInnerPanelLayout.createSequentialGroup()

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addComponent(carrierimageLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 120, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(watermarkLabel))
                            .addGap(8, 8, 8)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(carrierimageTextArea, javax.swing.GroupLayout.PREFERRED_SIZE, 178, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(watermarkTextArea, javax.swing.GroupLayout.PREFERRED_SIZE, 178, javax.swing.GroupLayout.PREFERRED_SIZE)))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, embed_imagesInnerPanelLayout.createSequentialGroup()
                            .addComponent(encryptionkeyLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 127, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(encryptionkeyTextArea, javax.swing.GroupLayout.PREFERRED_SIZE, 177, javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(watermark_browseButton, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addComponent(carrierimage_browseButton))

.addComponent(encryptionkey_setButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(carrierimage_clearButton)
                        .addComponent(watermark_clearButton)

.addComponent(encryptionkey_clearButton))))
                    .addGap(56, 56, 56)))
            .addContainerGap())
        );
        embed_imagesInnerPanelLayout.setVerticalGroup(

embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(embed_imagesInnerPanelLayout.createSequentialGroup()
            .addContainerGap()

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(carrierimage_browseButton)
                .addComponent(carrierimageLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(carrierimageTextArea, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(carrierimage_clearButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(watermark_browseButton, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(watermark_clearButton))
                .addComponent(watermarkTextArea, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(watermarkLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(encryptionkeyCheckBox)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(encryptionkeyLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(encryptionkeyTextArea, javax.swing.GroupLayout.PREFERRED_SIZE, 23, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(encryptionkey_clearButton)
                .addComponent(encryptionkey_setButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(embed_imagesInnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(embed_startButton)
                .addComponent(embed_clearButton))
            .addGap(55, 55, 55))
        );

        javax.swing.GroupLayout embed_imagesPanelLayout = new javax.swing.GroupLayout(embed_imagesPanel);

embed_imagesPanel.setLayout(embed_imagesPanelLayout);
        embed_imagesPanelLayout.setHorizontalGroup(

embed_imagesPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(embed_imagesInnerPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        embed_imagesPanelLayout.setVerticalGroup(

embed_imagesPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(embed_imagesInnerPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 154, javax.swing.GroupLayout.PREFERRED_SIZE)
        );

        javax.swing.GroupLayout embedTabLayout = new javax.swing.GroupLayout(embedTab);
        embedTab.setLayout(embedTabLayout);
        embedTabLayout.setHorizontalGroup(

embedTabLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, embedTabLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(embedTabLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(embed_imagesPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(embed_settingsPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
        embedTabLayout.setVerticalGroup(

embedTabLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

            .addGroup(embedTabLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(embed_settingsPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(embed_imagesPanel, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(90, 90, 90))
        );

        applicationTabbedPane.addTab("Embed Watermark", new javax.swing.ImageIcon(getClass().getResource("/my/reversiblewatermarking/embed.gif")), embedTab); // NOI18N

        extractTab.setBackground(new java.awt.Color(102, 0, 0));

        extract_settingsPanel.setBackground(new java.awt.Color(102, 0, 0));

extract_settingsPanel.setBorder(javax.swing.BorderFactory.createTitledBorder(new javax.swing.border.LineBorder(new java.awt.Color(255, 255, 255), 3, true), "Extraction and Restoration Settings", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION, javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma", 1, 14), new java.awt.Color(230, 230, 71))); // NOI18N

        extract_settingsInnerPanel.setBackground(new java.awt.Color(255, 255, 255));

        restoredwatermark_setButton.setText("Set");
        restoredwatermark_setButton.setOpaque(false);
        restoredwatermark_setButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                restoredwatermark_setButtonActionPerformed(evt);
            }
        });

        restoredimageLabel.setText("Original Image Filename");

        restoredpathTextArea.setEditable(false);
        restoredpathTextArea.setBackground(new java.awt.Color(255, 255, 255));

restoredpathTextArea.setText(master.extractor.get_filepath());
        restoredpathTextArea.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent evt) {
                restoredpathTextAreaMouseEntered(evt);
            }
        });

        restoredwatermarkLabel.setText("Watermark Filename");


restoredimageTextArea.setText(master.extractor.get_restored_fil
```

```
ename());
        restoredimageTextArea.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                restoredimageTextAreaMouseEntered(evt);
            }
        });
        restoredimageTextArea.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                restoredimageTextAreaActionPerformed(evt);
            }
        });

        def_restoredwatermarkButton.setText("Use Default
Filename");
        def_restoredwatermarkButton.setOpaque(false);
        def_restoredwatermarkButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                def_restoredwatermarkButtonActionPerformed(evt);
            }
        });

        def_restoredimageButton.setText("Use Default Filename");
        def_restoredimageButton.setOpaque(false);
        def_restoredimageButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                def_restoredimageButtonActionPerformed(evt);
            }
        });


restoredwatermarkTextArea.setText(master.extractor.get_waterm
ark_filename());
        restoredwatermarkTextArea.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                restoredwatermarkTextAreaMouseEntered(evt);
            }
        });
        restoredwatermarkTextArea.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                restoredwatermarkTextAreaActionPerformed(evt);
            }
        });

        restoredpath_defButton.setText("Use Default Path");
        restoredpath_defButton.setOpaque(false);
        restoredpath_defButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                restoredpath_defButtonActionPerformed(evt);
            }
        });

        restoredpathLabel.setText("File Directory");

        restoredimage_setButton.setText("Set");
        restoredimage_setButton.setOpaque(false);
        restoredimage_setButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                restoredimage_setButtonActionPerformed(evt);
            }
        });

        restoredpath_browseButton.setText("Browse");
        restoredpath_browseButton.setOpaque(false);
        restoredpath_browseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                restoredpath_browseButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout extract_settingsInnerPanelLayout
= new javax.swing.GroupLayout(extract_settingsInnerPanel);

extract_settingsInnerPanel.setLayout(extract_settingsInnerPanel
Layout);
        extract_settingsInnerPanelLayout.setHorizontalGroup(

extract_settingsInnerPanelLayout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(extract_settingsInnerPanelLayout.createSequentialG
roup()
                .addContainerGap()

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(extract_settingsInnerPanelLayout.createSequentialG
roup()
                        .addComponent(restoredimageLabel)
                        .addGap(18, 18, 18)
                        .addComponent(restoredimageTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 180,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(extract_settingsInnerPanelLayout.createSequentialG
roup()

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(restoredpathLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(restoredwatermarkLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 114,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(18, 18, 18)

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(restoredpathTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 180,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(restoredwatermarkTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 180,
javax.swing.GroupLayout.PREFERRED_SIZE))))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.TRAILING, false)
```

```
.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
extract_settingsInnerPanelLayout.createSequentialGroup()
                    .addComponent(restoredimage_setButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(def_restoredimageButton))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING,
extract_settingsInnerPanelLayout.createSequentialGroup()
                    .addComponent(restoredpath_browseButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                    .addComponent(restoredpath_defButton))

.addGroup(extract_settingsInnerPanelLayout.createSequentialG
roup()
                    .addComponent(restoredwatermark_setButton)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(def_restoredwatermarkButton)))
                .addContainerGap(29, Short.MAX_VALUE))
        );
        extract_settingsInnerPanelLayout.setVerticalGroup(

extract_settingsInnerPanelLayout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(extract_settingsInnerPanelLayout.createSequentialG
roup()
                .addContainerGap()

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(restoredimageLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(restoredimageTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(def_restoredimageButton)
                    .addComponent(restoredimage_setButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.BASELINE)

                    .addComponent(restoredwatermarkTextArea,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(def_restoredwatermarkButton,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(restoredwatermark_setButton))
                .addComponent(restoredwatermarkLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(6, 6, 6)

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(restoredpathLabel)

.addGroup(extract_settingsInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(restoredpathTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(restoredpath_browseButton)
                    .addComponent(restoredpath_defButton)))
                .addContainerGap())
        );

        javax.swing.GroupLayout extract_settingsPanelLayout =
new javax.swing.GroupLayout(extract_settingsPanel);

extract_settingsPanel.setLayout(extract_settingsPanelLayout);
        extract_settingsPanelLayout.setHorizontalGroup(

extract_settingsPanelLayout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.LEADING)
            .addComponent(extract_settingsInnerPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        extract_settingsPanelLayout.setVerticalGroup(

extract_settingsPanelLayout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.LEADING)
            .addComponent(extract_settingsInnerPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );


extract_imagesPanel.setBorder(javax.swing.BorderFactory.creat
eTitledBorder(new javax.swing.border.LineBorder(new
java.awt.Color(255, 255, 255), 3, true), "Watermark Extraction
and Image Restoration",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new
java.awt.Font("Tahoma", 1, 14), new java.awt.Color(230, 230,
71))); // NOI18N
        extract_imagesPanel.setOpaque(false);

        extract_imagesInnerPanel.setBackground(new
java.awt.Color(255, 255, 255));

        inputimageLabel.setText("Select Image(s):");

        decryptionkeyLabel.setText("Select Decryption Key:");

        decryptionkey_clearButton.setText("Clear");
        decryptionkey_clearButton.setEnabled(false);
        decryptionkey_clearButton.setOpaque(false);
        decryptionkey_clearButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                decryptionkey_clearButtonActionPerformed(evt);
            }
        });

        decryptionkeyTextArea.setEditable(false);
        decryptionkeyTextArea.setMaximumSize(new
java.awt.Dimension(6, 20));

        inputimage_clearButton.setText("Clear");
```

```
        inputimage_clearButton.setOpaque(false);
        inputimage_clearButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                inputimage_clearButtonActionPerformed(evt);
            }
        });

        inputimage_browseButton.setText("Browse Files");
        inputimage_browseButton.setOpaque(false);
        inputimage_browseButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                inputimage_browseButtonActionPerformed(evt);
            }
        });

        decryptionkeyCheckBox.setText("Decrypt Watermarked
Image?");
        decryptionkeyCheckBox.setOpaque(false);
        decryptionkeyCheckBox.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                decryptionkeyCheckBoxActionPerformed(evt);
            }
        });

        inputimageTextArea.setEditable(false);
        inputimageTextArea.setBackground(new
java.awt.Color(255, 255, 255));
        inputimageTextArea.setMaximumSize(new
java.awt.Dimension(6, 20));
        inputimageTextArea.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseEntered(java.awt.event.MouseEvent
evt) {
                inputimageTextAreaMouseEntered(evt);
            }
        });

        extract_startButton.setText("Extract Watermark");
        extract_startButton.setOpaque(false);
        extract_startButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                extract_startButtonActionPerformed(evt);
            }
        });

        extract_clearButton.setText("Clear Selections");
        extract_clearButton.setOpaque(false);
        extract_clearButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                extract_clearButtonActionPerformed(evt);
            }
        });

        decryptionkey_setButton.setText("Set Key");
        decryptionkey_setButton.setEnabled(false);
        decryptionkey_setButton.setOpaque(false);
        decryptionkey_setButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                decryptionkey_setButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout extract_imagesInnerPanelLayout
= new javax.swing.GroupLayout(extract_imagesInnerPanel);

extract_imagesInnerPanel.setLayout(extract_imagesInnerPanel
Layout);
        extract_imagesInnerPanelLayout.setHorizontalGroup(

extract_imagesInnerPanelLayout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(extract_imagesInnerPanelLayout.createSequentialGr
oup()
                .addContainerGap()

.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
extract_imagesInnerPanelLayout.createSequentialGroup()
                        .addGap(0, 0, Short.MAX_VALUE)
                        .addComponent(extract_startButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 138,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                        .addComponent(extract_clearButton))

.addGroup(extract_imagesInnerPanelLayout.createSequentialGr
oup()

.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(decryptionkeyCheckBox)

.addGroup(extract_imagesInnerPanelLayout.createSequentialGr
oup()

.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING, false)

.addGroup(extract_imagesInnerPanelLayout.createSequentialGr
oup()
                                    .addComponent(inputimageLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 120,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(10, 10, 10)
                                    .addComponent(inputimageTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 178,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
extract_imagesInnerPanelLayout.createSequentialGroup()
                                    .addComponent(decryptionkeyLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 126,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addComponent(decryptionkeyTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 178,
javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.UNRELATED)
```

```
.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(decryptionkey_setButton)

.addComponent(inputimage_browseButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.LEADING)
                        .addComponent(inputimage_clearButton)

.addComponent(decryptionkey_clearButton))))
                .addGap(0, 0, Short.MAX_VALUE)))
            .addContainerGap())
        );
        extract_imagesInnerPanelLayout.setVerticalGroup(

extract_imagesInnerPanelLayout.createParallelGroup(javax.swi
ng.GroupLayout.Alignment.LEADING)

.addGroup(extract_imagesInnerPanelLayout.createSequentialGr
oup()
            .addContainerGap()

.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(inputimage_browseButton)
                .addComponent(inputimageLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
 javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(inputimageTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(inputimage_clearButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.UNRELATED)
            .addComponent(decryptionkeyCheckBox)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)

.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(decryptionkeyLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(decryptionkeyTextArea,
javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(decryptionkey_clearButton)
                .addComponent(decryptionkey_setButton))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.UNRELATED)

.addGroup(extract_imagesInnerPanelLayout.createParallelGrou
p(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(extract_startButton)
                .addComponent(extract_clearButton))
            .addContainerGap())
        );

        javax.swing.GroupLayout extract_imagesPanelLayout =
new javax.swing.GroupLayout(extract_imagesPanel);
        extract_imagesPanel.setLayout(extract_imagesPanelLayout);
        extract_imagesPanelLayout.setHorizontalGroup(

extract_imagesPanelLayout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.LEADING)
            .addComponent(extract_imagesInnerPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        extract_imagesPanelLayout.setVerticalGroup(

extract_imagesPanelLayout.createParallelGroup(javax.swing.Gr
oupLayout.Alignment.LEADING)
            .addComponent(extract_imagesInnerPanel,
javax.swing.GroupLayout.DEFAULT_SIZE, 131,
Short.MAX_VALUE)
        );

        javax.swing.GroupLayout extractTabLayout = new
javax.swing.GroupLayout(extractTab);
        extractTab.setLayout(extractTabLayout);
        extractTabLayout.setHorizontalGroup(

extractTabLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(extractTabLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(extractTabLayout.createParallelGroup(javax.swing.G
roupLayout.Alignment.LEADING)
                    .addComponent(extract_settingsPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(extract_imagesPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                .addContainerGap())
        );
        extractTabLayout.setVerticalGroup(

extractTabLayout.createParallelGroup(javax.swing.GroupLayout.
Alignment.LEADING)
            .addGroup(extractTabLayout.createSequentialGroup()
                .addContainerGap()
                .addComponent(extract_settingsPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(extract_imagesPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(43, 43, 43))
        );

        applicationTabbedPane.addTab("Extract Watermark", new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/extracticon.gif")), extractTab); // NOI18N

        helpTab.setBackground(new java.awt.Color(102, 0, 0));
        helpTab.setMaximumSize(new java.awt.Dimension(579,
322));
        helpTab.setMinimumSize(new java.awt.Dimension(579,
322));

        helptext_Label.setFont(new java.awt.Font("Garamond", 1,
```

```
24)); // NOI18N
        helptext_Label.setForeground(new java.awt.Color(230,
230, 71));
        helptext_Label.setText("Help Topics");

        help_InnerPanel.setBackground(new java.awt.Color(255,
255, 255));

        helpinst_Label.setFont(new java.awt.Font("Tahoma", 0,
14)); // NOI18N
        helpinst_Label.setText("<html>This page contains more
information about the application. To select, just click on your
selected topic.<br><hr></html>");

        helpA_Label.setFont(new java.awt.Font("Tahoma", 0,
12)); // NOI18N
        helpA_Label.setText("<html><b>A. Reversible
Watermarking</b><br><blockquote>A watermarking procedure
that allows perfect restoration of the original image upon
watermark extraction.</blockquote></html>");
        helpA_Label.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
                helpA_LabelMouseClicked(evt);
            }
        });

        helpB_Label.setFont(new java.awt.Font("Tahoma", 0,
12)); // NOI18N
        helpB_Label.setText("<html><b>B. Congruence
Computations</b><br><blockquote>A reversible watermarking
algorithm that relies on congruence computations to achieve
reversibility.</blockquote></html>");
        helpB_Label.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
                helpB_LabelMouseClicked(evt);
            }
        });

        helpC_Label.setFont(new java.awt.Font("Tahoma", 0,
12)); // NOI18N
        helpC_Label.setText("<html><b>C. Encryption using the
Advanced Encryption
Standard(AES)</b><br><blockquote>More information on how
Java handles file encryption, and about the AES
algorithm.</blockquote></html>");
        helpC_Label.addMouseListener(new
java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent
evt) {
                helpC_LabelMouseClicked(evt);
            }
        });

        javax.swing.GroupLayout help_InnerPanelLayout = new
javax.swing.GroupLayout(help_InnerPanel);
        help_InnerPanel.setLayout(help_InnerPanelLayout);
        help_InnerPanelLayout.setHorizontalGroup(

help_InnerPanelLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(help_InnerPanelLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(help_InnerPanelLayout.createParallelGroup(javax.sw
ing.GroupLayout.Alignment.LEADING)
                    .addComponent(helpinst_Label,
javax.swing.GroupLayout.DEFAULT_SIZE, 539,
Short.MAX_VALUE)
                    .addComponent(helpA_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)
                    .addComponent(helpB_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)
                    .addComponent(helpC_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE))
                .addContainerGap())
        );
        help_InnerPanelLayout.setVerticalGroup(

help_InnerPanelLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(help_InnerPanelLayout.createSequentialGroup()
                .addGap(6, 6, 6)
                .addComponent(helpinst_Label,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(helpA_Label,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(helpB_Label,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(helpC_Label,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(89, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout helpTabLayout = new
javax.swing.GroupLayout(helpTab);
        helpTab.setLayout(helpTabLayout);
        helpTabLayout.setHorizontalGroup(

helpTabLayout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(helpTabLayout.createSequentialGroup()
                .addContainerGap()

.addGroup(helpTabLayout.createParallelGroup(javax.swing.Gro
upLayout.Alignment.LEADING)
                    .addComponent(help_InnerPanel,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(helptext_Label,
javax.swing.GroupLayout.DEFAULT_SIZE, 559,
Short.MAX_VALUE))
                .addContainerGap())
        );
```

```
        helpTabLayout.setVerticalGroup(

helpTabLayout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(helpTabLayout.createSequentialGroup()
                .addGap(6, 6, 6)
                .addComponent(helptext_Label,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(help_InnerPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );

        applicationTabbedPane.addTab("Help Topics", new
javax.swing.ImageIcon(getClass().getResource("/my/reversiblew
atermarking/help.png")), helpTab);
 // NOI18N

        exitapplicationButton.setText("Exit Application");
        exitapplicationButton.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                exitapplicationButtonActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout applicationPanelLayout = new
javax.swing.GroupLayout(applicationPanel);
        applicationPanel.setLayout(applicationPanelLayout);
        applicationPanelLayout.setHorizontalGroup(

applicationPanelLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)
            .addComponent(applicationTabbedPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
applicationPanelLayout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(exitapplicationButton)
                .addContainerGap())
        );
        applicationPanelLayout.setVerticalGroup(

applicationPanelLayout.createParallelGroup(javax.swing.GroupL
ayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
applicationPanelLayout.createSequentialGroup()
                .addComponent(applicationTabbedPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 352,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                .addComponent(exitapplicationButton)
                .addGap(6, 6, 6))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addComponent(titlePanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(applicationPanel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(titlePanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(applicationPanel,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void
exitapplicationButtonActionPerformed(java.awt.event.ActionEve
nt evt) {//GEN-
FIRST:event_exitapplicationButtonActionPerformed
        System.exit(0);
    }//GEN-LAST:event_exitapplicationButtonActionPerformed

    private void
def_outimageButtonActionPerformed(java.awt.event.ActionEven
t evt) {//GEN-FIRST:event_def_outimageButtonActionPerformed

master.embedder.set_filename(master.embedder.WATERMARK
ED_DEF);

outimageTextArea.setText(master.embedder.get_filename());
    }//GEN-LAST:event_def_outimageButtonActionPerformed

    private void
def_restoredimageButtonActionPerformed(java.awt.event.Action
Event evt) {//GEN-
FIRST:event_def_restoredimageButtonActionPerformed

restoredimageTextArea.setText(master.extractor.RESTORED_D
EF);

master.extractor.set_restored_filename(master.extractor.WATER
MARK_DEF);
    }//GEN-
LAST:event_def_restoredimageButtonActionPerformed

    private void
inputimage_browseButtonActionPerformed(java.awt.event.Actio
nEvent evt) {//GEN-
FIRST:event_inputimage_browseButtonActionPerformed
        JFileChooser imageChooser = new JFileChooser();
        imageChooser.setMultiSelectionEnabled(true);
```

```
        int result =  imageChooser.showOpenDialog(null);
        switch(result){
            case JFileChooser.APPROVE_OPTION:   File[]
imageFile = imageChooser.getSelectedFiles();
                                    String filename = "";
                                    for(int i=0; i<imageFile.length; i++)
{
                                        filename +=
(imageFile[i].getAbsolutePath() + "; ");
                                    }

inputimageTextArea.setText(filename);

master.set_inputImages(imageFile);
                                    break;
            case JFileChooser.CANCEL_OPTION:    break;
            case JFileChooser.ERROR_OPTION:     break;
        }
    }//GEN-
LAST:event_inputimage_browseButtonActionPerformed

    private void
decryptionkeyCheckBoxActionPerformed(java.awt.event.ActionE
vent evt) {//GEN-
FIRST:event_decryptionkeyCheckBoxActionPerformed
        if(decryptionkeyCheckBox.isSelected()){
            decryptionkeyTextArea.setEditable(true);
            decryptionkey_setButton.setEnabled(true);
            decryptionkey_clearButton.setEnabled(true);
        }
        else{
            decryptionkeyTextArea.setText("");
            master.set_decryptionKey((String)null);
            decryptionkeyTextArea.setEditable(false);
            decryptionkey_setButton.setEnabled(false);
            decryptionkey_clearButton.setEnabled(false);
        }
    }//GEN-LAST:event_decryptionkeyCheckBoxActionPerformed

    private void
decryptionkey_browseButtonActionPerformed(java.awt.event.Ac
tionEvent evt) {//GEN-
FIRST:event_decryptionkey_browseButtonActionPerformed
        JFileChooser decryptKeyChooser = new JFileChooser();
        int result =  decryptKeyChooser.showOpenDialog(null);
        switch(result){
            case JFileChooser.APPROVE_OPTION:   File
decryptKeyFile = decryptKeyChooser.getSelectedFile();
                                    String filename =
decryptKeyFile.getAbsolutePath();

decryptionkeyTextArea.setText(filename);

master.set_decryptionKey((String)null);//decryptKeyFile);
                                    break;
            case JFileChooser.CANCEL_OPTION:    break;
            case JFileChooser.ERROR_OPTION:     break;
        }
    }//GEN-
LAST:event_decryptionkey_browseButtonActionPerformed

    private void
encryptionkey_browseButtonActionPerformed(java.awt.event.Ac
tionEvent evt) {//GEN-
FIRST:event_encryptionkey_browseButtonActionPerformed
        JFileChooser encryptKeyChooser = new JFileChooser();
        int result =  encryptKeyChooser.showOpenDialog(null);
        switch(result){
            case JFileChooser.APPROVE_OPTION:   File
encryptKeyFile = encryptKeyChooser.getSelectedFile();
                                    String filename =
encryptKeyFile.getAbsolutePath();

encryptionkeyTextArea.setText(filename);

master.set_encryptionKey((String)null);//encryptKeyFile);
                                    break;
            case JFileChooser.CANCEL_OPTION:    break;
            case JFileChooser.ERROR_OPTION:     break;
        }
    }//GEN-
LAST:event_encryptionkey_browseButtonActionPerformed

    private void
extract_startButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_extract_startButtonActionPerformed
        Sleeper task = new Sleeper(1);
        task.execute();

master.extractor.set_restored_filename(restoredimageTextArea.
getText());
        this.inputimageTextArea.setText("");
        //this.decryptionkeyTextArea.setText("");
    }//GEN-LAST:event_extract_startButtonActionPerformed

    private void
inputimage_clearButtonActionPerformed(java.awt.event.ActionE
vent evt) {//GEN-
FIRST:event_inputimage_clearButtonActionPerformed
        inputimageTextArea.setText("");
        master.set_inputImages(null);
    }//GEN-LAST:event_inputimage_clearButtonActionPerformed

    private void
decryptionkey_clearButtonActionPerformed(java.awt.event.Actio
nEvent evt) {//GEN-
FIRST:event_decryptionkey_clearButtonActionPerformed
        decryptionkeyTextArea.setText("");
        master.set_decryptionKey((String)null);
    }//GEN-
LAST:event_decryptionkey_clearButtonActionPerformed

    private void
extract_clearButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_extract_clearButtonActionPerformed
        this.inputimageTextArea.setText("");
        this.decryptionkeyTextArea.setText("");
        master.clear_input();
    }//GEN-LAST:event_extract_clearButtonActionPerformed

    private void
outpath_browseButtonActionPerformed(java.awt.event.ActionEv
ent evt) {//GEN-
FIRST:event_outpath_browseButtonActionPerformed
        JFileChooser outpathChooser = new JFileChooser();

outpathChooser.setFileSelectionMode(JFileChooser.DIRECTOR
IES_ONLY);
        int result =  outpathChooser.showOpenDialog(null);
        switch(result){
            case JFileChooser.APPROVE_OPTION:   File
outpathFile = outpathChooser.getSelectedFile();
                                    String filename =
outpathFile.getAbsolutePath();

outpathTextArea.setText(filename);

master.embedder.set_filepath(filename);
                                    break;
            case JFileChooser.CANCEL_OPTION:    break;
```

```
            case JFileChooser.ERROR_OPTION:     break;
        }
    }//GEN-LAST:event_outpath_browseButtonActionPerformed

    private void
outpath_defButtonActionPerformed(java.awt.event.ActionEvent
 evt) {//GEN-FIRST:event_outpath_defButtonActionPerformed

master.embedder.set_filepath(System.getProperty("user.dir"));
        outpathTextArea.setText(master.embedder.get_filepath());
    }//GEN-LAST:event_outpath_defButtonActionPerformed

    private void
restoredpath_browseButtonActionPerformed(java.awt.event.Acti
onEvent evt) {//GEN-
FIRST:event_restoredpath_browseButtonActionPerformed
        JFileChooser restoredpathChooser = new JFileChooser();

restoredpathChooser.setFileSelectionMode(JFileChooser.DIRE
CTORIES_ONLY);
        int result =  restoredpathChooser.showOpenDialog(null);
        switch(result){
            case JFileChooser.APPROVE_OPTION:   File
restoredpathFile = restoredpathChooser.getSelectedFile();
                                    String filename =
restoredpathFile.getAbsolutePath();

restoredpathTextArea.setText(filename);

master.extractor.set_filepath(filename);
                                    break;
            case JFileChooser.CANCEL_OPTION:    break;
            case JFileChooser.ERROR_OPTION:     break;
        }
    }//GEN-
LAST:event_restoredpath_browseButtonActionPerformed

    private void
restoredpath_defButtonActionPerformed(java.awt.event.ActionE
vent evt) {//GEN-
FIRST:event_restoredpath_defButtonActionPerformed

restoredpathTextArea.setText(System.getProperty("user.dir"));

master.extractor.set_filepath(System.getProperty("user.dir"));
    }//GEN-LAST:event_restoredpath_defButtonActionPerformed

    private void
def_restoredwatermarkButtonActionPerformed(java.awt.event.A
ctionEvent evt) {//GEN-
FIRST:event_def_restoredwatermarkButtonActionPerformed

restoredwatermarkTextArea.setText(master.extractor.WATERMA
RK_DEF);

master.extractor.set_watermark_filename(master.extractor.WAT
ERMARK_DEF);
    }//GEN-
LAST:event_def_restoredwatermarkButtonActionPerformed

    private void
outimage_setButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_outimage_setButtonActionPerformed
        String input = outimageTextArea.getText();
        if(input.equalsIgnoreCase("")){

master.embedder.set_filename(master.embedder.WATERMARK
ED_DEF);

outimageTextArea.setText(master.embedder.WATERMARKED_
DEF);
        }
        else{
            master.embedder.set_filename(input);
        }
    }//GEN-LAST:event_outimage_setButtonActionPerformed

    private void
outimageTextAreaActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_outimageTextAreaActionPerformed
        String input = outimageTextArea.getText();
        if(input.equalsIgnoreCase("")){

master.embedder.set_filename(master.embedder.WATERMARK
ED_DEF);

outimageTextArea.setText(master.embedder.WATERMARKED_
DEF);
        }
        else{
            master.embedder.set_filename(input);
        }
    }//GEN-LAST:event_outimageTextAreaActionPerformed

    private void
outpathTextAreaActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_outpathTextAreaActionPerformed

    }//GEN-LAST:event_outpathTextAreaActionPerformed

    private void
restoredimageTextAreaActionPerformed(java.awt.event.ActionE
vent evt) {//GEN-
FIRST:event_restoredimageTextAreaActionPerformed
        String input = restoredimageTextArea.getText();
        if(input.equalsIgnoreCase("")){

restoredimageTextArea.setText(master.extractor.RESTORED_D
EF);

master.extractor.set_restored_filename(master.extractor.RESTO
RED_DEF);
        }
        else{
            master.extractor.set_restored_filename(input);
        }
    }//GEN-LAST:event_restoredimageTextAreaActionPerformed

    private void
restoredimage_setButtonActionPerformed(java.awt.event.Action
Event evt) {//GEN-
FIRST:event_restoredimage_setButtonActionPerformed
        String input = restoredimageTextArea.getText();
        if(input.equalsIgnoreCase("")){

restoredimageTextArea.setText(master.extractor.RESTORED_D
EF);

master.extractor.set_restored_filename(master.extractor.RESTO
RED_DEF);
        }
        else{
            master.extractor.set_restored_filename(input);
        }
    }//GEN-
LAST:event_restoredimage_setButtonActionPerformed

    private void
restoredwatermarkTextAreaActionPerformed(java.awt.event.Acti
onEvent evt) {//GEN-
```

```
FIRST:event_restoredwatermarkTextAreaActionPerformed
        String input = restoredwatermarkTextArea.getText();
        if(input.equalsIgnoreCase("")){

restoredwatermarkTextArea.setText(master.extractor.WATERMA
RK_DEF);

master.extractor.set_watermark_filename(master.extractor.WAT
ERMARK_DEF);
        }
        else{
            master.extractor.set_watermark_filename(input);
        }
    }//GEN-
LAST:event_restoredwatermarkTextAreaActionPerformed

    private void
restoredwatermark_setButtonActionPerformed(java.awt.event.A
ctionEvent evt) {//GEN-
FIRST:event_restoredwatermark_setButtonActionPerformed
        String input = restoredwatermarkTextArea.getText();
        if(input.equalsIgnoreCase("")){

restoredwatermarkTextArea.setText(master.extractor.WATERMA
RK_DEF);

master.extractor.set_watermark_filename(master.extractor.WAT
ERMARK_DEF);
        }
        else{
            master.extractor.set_watermark_filename(input);
        }
    }//GEN-
LAST:event_restoredwatermark_setButtonActionPerformed

    private void
outimageTextAreaMouseEntered(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_outimageTextAreaMouseEntered

outimageTextArea.setToolTipText(outimageTextArea.getText());
    }//GEN-LAST:event_outimageTextAreaMouseEntered

    private void
outpathTextAreaMouseEntered(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_outpathTextAreaMouseEntered
        outpathTextArea.setToolTipText(outpathTextArea.getText());
    }//GEN-LAST:event_outpathTextAreaMouseEntered

    private void
restoredimageTextAreaMouseEntered(java.awt.event.MouseEve
nt evt) {//GEN-
FIRST:event_restoredimageTextAreaMouseEntered

restoredimageTextArea.setToolTipText(restoredimageTextArea.g
etText());
    }//GEN-LAST:event_restoredimageTextAreaMouseEntered

    private void
restoredwatermarkTextAreaMouseEntered(java.awt.event.Mous
eEvent evt) {//GEN-
FIRST:event_restoredwatermarkTextAreaMouseEntered

restoredwatermarkTextArea.setToolTipText(restoredwatermarkTe
xtArea.getText());
    }//GEN-
LAST:event_restoredwatermarkTextAreaMouseEntered

    private void
restoredpathTextAreaMouseEntered(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_restoredpathTextAreaMouseEntered

restoredpathTextArea.setToolTipText(restoredpathTextArea.getT
ext());
    }//GEN-LAST:event_restoredpathTextAreaMouseEntered

    private void
inputimageTextAreaMouseEntered(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_inputimageTextAreaMouseEntered

inputimageTextArea.setToolTipText(inputimageTextArea.getText(
));
    }//GEN-LAST:event_inputimageTextAreaMouseEntered

    private void
decryptionkey_setButtonActionPerformed(java.awt.event.Action
Event evt) {//GEN-
FIRST:event_decryptionkey_setButtonActionPerformed
        String password = decryptionkeyTextArea.getText();
        master.set_decryptionKey(password);
    }//GEN-LAST:event_decryptionkey_setButtonActionPerformed

    private void
embed_clearButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_embed_clearButtonActionPerformed
        this.carrierimageTextArea.setText("");
        this.decryptionkeyTextArea.setText("");
        this.watermarkTextArea.setText("");
        master.clear_input();
    }//GEN-LAST:event_embed_clearButtonActionPerformed

    private void
embed_startButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_embed_startButtonActionPerformed
        Sleeper task = new Sleeper(0);
        task.execute();

master.embedder.set_filename(outimageTextArea.getText());
        this.carrierimageTextArea.setText("");
        //this.encryptionkeyTextArea.setText("");
        this.watermarkTextArea.setText("");
    }//GEN-LAST:event_embed_startButtonActionPerformed

    private void
encryptionkeyCheckBoxActionPerformed(java.awt.event.ActionE
vent evt) {//GEN-
FIRST:event_encryptionkeyCheckBoxActionPerformed
        if(encryptionkeyCheckBox.isSelected()){
            encryptionkeyTextArea.setEditable(true);
            encryptionkey_setButton.setEnabled(true);
            encryptionkey_clearButton.setEnabled(true);
        }
        else{
            encryptionkeyTextArea.setText("");
            master.set_encryptionKey((String)null);
            encryptionkeyTextArea.setEditable(false);
            encryptionkey_setButton.setEnabled(false);
            encryptionkey_clearButton.setEnabled(false);
        }
    }//GEN-LAST:event_encryptionkeyCheckBoxActionPerformed

    private void
encryptionkeyTextAreaMouseEntered(java.awt.event.MouseEve
nt evt) {//GEN-
FIRST:event_encryptionkeyTextAreaMouseEntered

encryptionkeyTextArea.setToolTipText(encryptionkeyTextArea.ge
tText());
    }//GEN-LAST:event_encryptionkeyTextAreaMouseEntered

    private void
```

```
encryptionkey_clearButtonActionPerformed(java.awt.event.Actio
nEvent evt) {//GEN-
FIRST:event_encryptionkey_clearButtonActionPerformed
        encryptionkeyTextArea.setText("");
        master.set_encryptionKey((String)null);
    }//GEN-
LAST:event_encryptionkey_clearButtonActionPerformed

    private void
encryptionkey_setButtonActionPerformed(java.awt.event.Action
Event evt) {//GEN-
FIRST:event_encryptionkey_setButtonActionPerformed
        String password = encryptionkeyTextArea.getText();
        master.set_encryptionKey(password);
    }//GEN-LAST:event_encryptionkey_setButtonActionPerformed

    private void
carrierimage_clearButtonActionPerformed(java.awt.event.Action
Event evt) {//GEN-
FIRST:event_carrierimage_clearButtonActionPerformed
        carrierimageTextArea.setText("");

        master.set_carrierImages(null);
    }//GEN-
LAST:event_carrierimage_clearButtonActionPerformed

    private void
watermark_clearButtonActionPerformed(java.awt.event.ActionEv
ent evt) {//GEN-
FIRST:event_watermark_clearButtonActionPerformed
        watermarkTextArea.setText("");
        master.set_watermark(null);
    }//GEN-LAST:event_watermark_clearButtonActionPerformed

    private void
watermark_browseButtonActionPerformed(java.awt.event.Action
Event evt) {//GEN-
FIRST:event_watermark_browseButtonActionPerformed
        JFileChooser watermarkChooser = new JFileChooser();
        int result = watermarkChooser.showOpenDialog(null);
        switch(result){
            case JFileChooser.APPROVE_OPTION:   File
watermarkFile = watermarkChooser.getSelectedFile();
            String filename = watermarkFile.getAbsolutePath();
            watermarkTextArea.setText(filename);
            master.set_watermark(watermarkFile);
            break;
            case JFileChooser.CANCEL_OPTION:    break;
            case JFileChooser.ERROR_OPTION:     break;
        }
    }//GEN-
LAST:event_watermark_browseButtonActionPerformed

    private void
carrierimage_browseButtonActionPerformed(java.awt.event.Acti
onEvent evt) {//GEN-
FIRST:event_carrierimage_browseButtonActionPerformed
        JFileChooser carrierChooser = new JFileChooser();
        carrierChooser.setMultiSelectionEnabled(true);
        int result = carrierChooser.showOpenDialog(null);
        switch(result){
            case JFileChooser.APPROVE_OPTION:   File[]
carrierFile = carrierChooser.getSelectedFiles();
            String filename = "";
            for(int i=0; i<carrierFile.length; i++){
                filename += (carrierFile[i].getAbsolutePath() + "; ");
            }
            carrierimageTextArea.setText(filename);
            master.set_carrierImages(carrierFile);
            break;
            case JFileChooser.CANCEL_OPTION:    break;
            case JFileChooser.ERROR_OPTION:     break;
        }
    }//GEN-
LAST:event_carrierimage_browseButtonActionPerformed

    private void
carrierimageTextAreaMouseEntered(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_carrierimageTextAreaMouseEntered

carrierimageTextArea.setToolTipText(carrierimageTextArea.getTe
xt());
    }//GEN-LAST:event_carrierimageTextAreaMouseEntered

    private void
watermarkTextAreaMouseEntered(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_watermarkTextAreaMouseEntered

watermarkTextArea.setToolTipText(watermarkTextArea.getText())
;
    }//GEN-LAST:event_watermarkTextAreaMouseEntered

    private void
instructionsdesc_LabelMouseClicked(java.awt.event.MouseEve
nt evt) {//GEN-
FIRST:event_instructionsdesc_LabelMouseClicked
        applicationTabbedPane.setSelectedIndex(1);
    }//GEN-LAST:event_instructionsdesc_LabelMouseClicked

    private void
embeddesc_LabelMouseClicked(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_embeddesc_LabelMouseClicked
        applicationTabbedPane.setSelectedIndex(2);
    }//GEN-LAST:event_embeddesc_LabelMouseClicked

    private void
extractdesc_LabelMouseClicked(java.awt.event.MouseEvent
evt) {//GEN-FIRST:event_extractdesc_LabelMouseClicked
        applicationTabbedPane.setSelectedIndex(3);
    }//GEN-LAST:event_extractdesc_LabelMouseClicked

    private void
helpdesc_LabelMouseClicked(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_helpdesc_LabelMouseClicked
        applicationTabbedPane.setSelectedIndex(4);
    }//GEN-LAST:event_helpdesc_LabelMouseClicked

    private void
helpA_LabelMouseClicked(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_helpA_LabelMouseClicked
        String helpinfo = "\tReversible Watermarking is a variation
on the traditional digital watermarking "
                + "procedure such that it allows the perfect restoration
of the original carrier image upon watermark "
                + "extraction. \n\n\tThe traditional process of digital
watermarking inevitably produces some "
                + "distortion to the carrier image used. These
distortions are made to be as small as possible, "
                + "however, these distortions are permanent and
irreversible. For some applications of images such as "
                + "in the military or the medical field, even small
changes in pixel values are unacceptable As such, "
                + "traditional watermarking is not sufficient and they
are when reversible watermarking is applicable."
                + "\n\n\tAs the need for reversible watermarking
increased, plenty of algorithms have been and are "
                + "being developed, such as Tian's algorithm. This
application uses the Congruence Computations "
                + "algorithm developed by Chaumont and Puech.";
        helpScreen helpA = new helpScreen();
```

```
        helpA.setVisible(true);
        helpA.jLabel1.setText("A. Reversible Watermarking");
        helpA.jTextArea1.setText(helpinfo);
    }//GEN-LAST:event_helpA_LabelMouseClicked

    private void
helpB_LabelMouseClicked(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_helpB_LabelMouseClicked
        String helpinfo = "\tThis reversible watermarking algorithm
was proposed by Chaumont and "
                + "Puech and relies on congruence computations to
achieve reversibility. This algorithm"
                + " has high embedding capacity along with a
relatively small complexity. After embedding"
                + "the watermark file, the visual quality of the image is
reduced, but since the image "
                + "will be restored after extraction of the watermark, it
is not considered to "
                + "be a problem. \n\n\tThe algorithm first defines a
constant integer value n "
                + "that is greater than or equal to 3(for this application,
n is given as 4), and the "
                + "T transform which takes on two integers x1 and x2
as input and returns an integer: "
                + "\n\n\t\tT(x1, x2) = (n + 1).x1 - n.x2 \n\n\tFor an
image I with size N(N = M x N pixels) "
                + "whose gray-levels belong to [0, L] where L = 2^B
and B is the number of grey levels of "
                + "the image. Each pixel i is then classified into three
possible states: "
                + "\n\t · Embedding: A pixel which contains a portion of
the watermark. \n\t· To-correct: "
                + "A pixel which has been modified but does not
embed any portion of the watermark. These "
                + "pixels will be corrected in the extraction process.
\n\t· Original: A pixel whose "
                + "value is unchanged. \n\n\tThe embedding process
is composed of two steps: "
                + "\n\t1. Classify each pixel in one of the three states:
embedding, to-correct, original, "
                + "\n\t2. Embed into the embedding pixels, the
watermark made of corrective codes plus the"
                + " watermark. \n\n\tNote that the image scan order
has been chosen to be from top to "
                + "bottom and from left to right. \n\n\tFor the decoding
process, the image scan order is inverted; "
                + "it is perform from bottom to top and from right to
left. The decoding process is also "
                + "composed of two steps : "
                + "\n\t1. Extract the watermark from the embedding
pixels, revert (during the scan) all those "
                + "pixels and localize the to-correct pixels, \n\t2. From
the extracted watermark retrieve the "
                + "corrective codes and the message, and correct the
to-correct pixels.";
        helpScreen helpB = new helpScreen();
        helpB.setVisible(true);
        helpB.jLabel1.setText("B. Congruence Computations");
        helpB.jTextArea1.setText(helpinfo);
    }//GEN-LAST:event_helpB_LabelMouseClicked

    private void
helpC_LabelMouseClicked(java.awt.event.MouseEvent evt)
{//GEN-FIRST:event_helpC_LabelMouseClicked
        String helpinfo = "\tEncryption is the process of converting
information to an encrypted form, "
                + "such that it is intelligible only to someone who
knows how to decrypt it in order to "
                + "obtain the original message. It involves taking data
that is called the 'plaintext' "
                + "and converting it to a 'ciphertext' using an
encryption algorithm and key. "
                + "\n\n\t Encryption is most often used to protect
information so that only an "
                + "authorized entity can access it. Some applications
include: \n\t"
                + "- Personal Use \n\t- Securing Networks \n\t- Access
Control"
                + "\n\n\tThe Advanced Encryption Standard(AES) is a
specification for the encryption "
                + "of electronic data that has been a standard in the
United States since 2001 as "
                + "US FIPS 197. Originally, it was called as 'Rijndael',
a combination of the names of"
                + "its developers, Joan Daemen and Vincent
Rijmen. \n\n\tThe AES processes sequences called "
                + "blocks each with length of 128 bits, along with a key
size of either 128, 192, "
                + "or 256 bits. This key size determines the number of
times the plaintext(data to be encrypted)"
                + " undergoes transformation in order to produce the
final output. These are 10, 12 and 14 "
                + "cycles of repetition for keys of size 128, 192 and
256 bits respectively. \n\n\tThe algorithm "
                + "used by the AES is classified as a symetric-key
algorithm. That is, the same key is used "
                + "for both encyption and decryption of the data.
\n\n\tAs for the security considerations on the "
                + "safety of the AES for use, it is considered as safe
from brute-force attacks due to the "
                + "incredibly large amount of time needed to crack the
algorithm. However, some attacks have "
                + "cracked the AES with a slightly improved time
 from the brute force attack. The said attack "
                + "however, would still take billions of years of
computer time, and thus the algorithm is still "
                + "considered as secure for now. \n\n\tThe application
uses the BouncyCastle Provider External Library "
                + "for Java and uses the
'PBEWITHSHA256AND128BITAES-CBC-BC' algorithm. The
user's password"
                + "is hashed using the Secure Hash Algorithm(SHA)
and generates an encryption key using the"
                + "hash and a series of bytes called the 'salt'. The
generated key is then used for the "
                + "128 bit AES algorithm.\n\n\tFor more information,
you can visit the following sites:\n"
                + "\t -http://www.bouncycastle.org/ \n\t
-http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf";
        helpScreen helpC = new helpScreen();
        helpC.setVisible(true);
        helpC.jLabel1.setText("C. Encryption using the AES");
        helpC.jTextArea1.setText(helpinfo);
    }//GEN-LAST:event_helpC_LabelMouseClicked

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/pl
af.html
         */
        try {
            javax.swing.UIManager.setLookAndFeel(
```

```
javax.swing.UIManager.getSystemLookAndFeelClassName());
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ReversibleWatermarkingUI.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ReversibleWatermarkingUI.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ReversibleWatermarkingUI.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException
ex) {

java.util.logging.Logger.getLogger(ReversibleWatermarkingUI.cl
ass.getName()).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ReversibleWatermarkingUI().setVisible(true);
            }
        });
    }
    private Application_handler master;
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JPanel applicationPanel;
    private javax.swing.JTabbedPane applicationTabbedPane;
    private javax.swing.JLabel carrierimageLabel;
    private javax.swing.JTextField carrierimageTextArea;
    private javax.swing.JButton carrierimage_browseButton;
    private javax.swing.JButton carrierimage_clearButton;
    private javax.swing.JCheckBox decryptionkeyCheckBox;
    private javax.swing.JLabel decryptionkeyLabel;
    private javax.swing.JTextField decryptionkeyTextArea;
    private javax.swing.JButton decryptionkey_browseButton;
    private javax.swing.JButton decryptionkey_clearButton;
    private javax.swing.JButton decryptionkey_setButton;
    private javax.swing.JButton def_outimageButton;
    private javax.swing.JButton def_restoredimageButton;
    private javax.swing.JButton def_restoredwatermarkButton;
    private javax.swing.JPanel embedTab;
    private javax.swing.JButton embed_clearButton;
    private javax.swing.JPanel embed_imagesInnerPanel;
    private javax.swing.JPanel embed_imagesPanel;
    private javax.swing.JLabel embed_inst1;
    private javax.swing.JLabel embed_inst2;
    private javax.swing.JLabel embed_inst3;
    private javax.swing.JLabel embed_inst4;
    private javax.swing.JLabel embed_inst5;
    private javax.swing.JLabel embed_inst6;
    private javax.swing.JPanel embed_instructionsPanel;
    private javax.swing.JPanel embed_settingInnerPanel;
    private javax.swing.JPanel embed_settingsPanel;
    private javax.swing.JButton embed_startButton;
    private javax.swing.JLabel embeddesc_Label;
    private javax.swing.JCheckBox encryptionkeyCheckBox;
    private javax.swing.JLabel encryptionkeyLabel;
    private javax.swing.JTextField encryptionkeyTextArea;
    private javax.swing.JButton encryptionkey_browseButton;
    private javax.swing.JButton encryptionkey_clearButton;
    private javax.swing.JButton encryptionkey_setButton;
    private javax.swing.JButton exitapplicationButton;
    private javax.swing.JPanel extractTab;
    private javax.swing.JButton extract_clearButton;
    private javax.swing.JPanel extract_imagesInnerPanel;
    private javax.swing.JPanel extract_imagesPanel;
    private javax.swing.JLabel extract_inst1;
    private javax.swing.JLabel extract_inst2;
    private javax.swing.JLabel extract_inst3;
    private javax.swing.JLabel extract_inst4;
    private javax.swing.JLabel extract_inst5;
    private javax.swing.JPanel extract_instructionsPanel;
    private javax.swing.JPanel extract_settingsInnerPanel;
    private javax.swing.JPanel extract_settingsPanel;
    private javax.swing.JButton extract_startButton;
    private javax.swing.JLabel extractdesc_Label;
    private javax.swing.JLabel helpA_Label;
    private javax.swing.JLabel helpB_Label;
    private javax.swing.JLabel helpC_Label;
    private javax.swing.JPanel helpTab;
    private javax.swing.JPanel help_InnerPanel;
    private javax.swing.JLabel helpdesc_Label;
    private javax.swing.JLabel helpinst_Label;
    private javax.swing.JLabel helptext_Label;
    private javax.swing.JLabel inputimageLabel;
    private javax.swing.JTextField inputimageTextArea;
    private javax.swing.JButton inputimage_browseButton;
    private javax.swing.JButton inputimage_clearButton;
    private javax.swing.JPanel instructTab;
    private javax.swing.JLabel instructionsdesc_Label;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JTextField outimageTextArea;
    private javax.swing.JButton outimage_setButton;
    private javax.swing.JLabel outimagenameLabel;
    private javax.swing.JLabel outpathLabel;
    private javax.swing.JTextField outpathTextArea;
    private javax.swing.JButton outpath_browseButton;
    private javax.swing.JButton outpath_defButton;
    private javax.swing.JLabel restoredimageLabel;
    private javax.swing.JTextField restoredimageTextArea;
    private javax.swing.JButton restoredimage_setButton;
    private javax.swing.JLabel restoredpathLabel;
    private javax.swing.JTextField restoredpathTextArea;
    private javax.swing.JButton restoredpath_browseButton;
    private javax.swing.JButton restoredpath_defButton;
    private javax.swing.JLabel restoredwatermarkLabel;
    private javax.swing.JTextField restoredwatermarkTextArea;
    private javax.swing.JButton restoredwatermark_setButton;
    private javax.swing.JPanel titlePanel;
    private javax.swing.JLabel watermarkLabel;
    private javax.swing.JTextField watermarkTextArea;
    private javax.swing.JButton watermark_browseButton;
    private javax.swing.JButton watermark_clearButton;
    private javax.swing.JPanel welcomeTab;
    private javax.swing.JPanel welcomeTab_InnerPanel;
    private javax.swing.JLabel welcomedesc_Label;
    private javax.swing.JLabel wlecometext_Label;
    // End of variables declaration//GEN-END:variables

    class Sleeper extends SwingWorker{
        private int setting;
        /*
         * 0 = embed watermark
         * 1 = extract watermark
         */

        Sleeper(int setting){
            this.setting = setting;
        }

        @Override
        public Void doInBackground() throws Exception {
            try{
                resultScreen s = new resultScreen();
```

```
            switch(this.setting){
               case 0: master.embed(s);
                     break;

               case 1: master.extract(s);
                     break;

               default: break;
            }

         }
         catch(Exception e){
            //Error handling
         }
         return null;
      }

   }
}

class BackgroundPanel extends JPanel
{
   Image image;
   public BackgroundPanel()
   {
   try
   {
      image = (new
ImageIcon(getClass().getResource("oblebg.jpg"))).getImage();
   }
   catch(Exception e){/*handled in paintComponent()*/}
   }

   public void paintComponent(Graphics g)
   {
   super.paintComponent(g);
      g.drawImage(image, 0, 0, null);
   }
}


package my.reversiblewatermarking;

/**
 * @author Dan Pantano
 */
public class resultScreen extends javax.swing.JFrame {

   public void update(String s) {
      resultTextArea.setText(resultTextArea.getText() + "\n" +s);
   }
   /**
    * Creates new form resultScreen
    */
   public resultScreen() {
      initComponents();
   }

   /**
    * This method is called from within the constructor to initialize
the form.
    * WARNING: Do NOT modify this code. The content of this
method is always
    * regenerated by the Form Editor.
    */
   @SuppressWarnings("unchecked")
   // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
   private void initComponents() {
      jPanel1 = new javax.swing.JPanel();
      resultScrollPane = new javax.swing.JScrollPane();
      resultTextArea = new javax.swing.JTextArea();
      exit_resultButton = new javax.swing.JButton();
      resultLabel = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPO
SE_ON_CLOSE);

      jPanel1.setBackground(new java.awt.Color(102, 0, 0));


resultScrollPane.setHorizontalScrollBarPolicy(javax.swing.Scroll
PaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

      resultTextArea.setEditable(false);
      resultTextArea.setColumns(20);
      resultTextArea.setLineWrap(true);
      resultTextArea.setRows(5);
      resultTextArea.setWrapStyleWord(true);
      resultScrollPane.setViewportView(resultTextArea);

      exit_resultButton.setText("Close Window");
      exit_resultButton.setEnabled(false);
      exit_resultButton.addActionListener(new
java.awt.event.ActionListener() {
         public void actionPerformed(java.awt.event.ActionEvent
evt) {
            exit_resultButtonActionPerformed(evt);
         }
      });

      resultLabel.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
      resultLabel.setForeground(new java.awt.Color(230, 230,
71));
      resultLabel.setText("Application Status:");

      javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
      jPanel1.setLayout(jPanel1Layout);
      jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
         .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                     .addGap(150, 150, 150)
                     .addComponent(exit_resultButton)
                     .addGap(132, 132, 132))
                  .addGroup(jPanel1Layout.createSequentialGroup()
                     .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)
                        .addComponent(resultScrollPane)
                        .addComponent(resultLabel,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))
            .addContainerGap())
      );
      jPanel1Layout.setVerticalGroup(
```

```
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(resultLabel,
javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(resultScrollPane,
javax.swing.GroupLayout.PREFERRED_SIZE, 225,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(exit_resultButton)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addComponent(jPanel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addComponent(jPanel1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void
exit_resultButtonActionPerformed(java.awt.event.ActionEvent
evt) {//GEN-FIRST:event_exit_resultButtonActionPerformed
        this.dispose();
    }//GEN-LAST:event_exit_resultButtonActionPerformed

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/pl
af.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(resultScreen.class.getName()
).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(resultScreen.class.getName()
).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(resultScreen.class.getName()
).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException
ex) {

java.util.logging.Logger.getLogger(resultScreen.class.getName()
).log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new resultScreen().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    protected javax.swing.JButton exit_resultButton;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JLabel resultLabel;
    private javax.swing.JScrollPane resultScrollPane;
    private javax.swing.JTextArea resultTextArea;
    // End of variables declaration//GEN-END:variables
}



package my.reversiblewatermarking;

/**
 * @author Dan Pantano
 */
public class helpScreen extends javax.swing.JFrame {

    /**
     * Creates new form helpScreen
     */
    public helpScreen() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize
the form.
     * WARNING: Do NOT modify this code. The content of this
method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
```

```
        jButton1 = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.DISPO
SE_ON_CLOSE);
        setMaximumSize(new java.awt.Dimension(579, 322));
        setMinimumSize(new java.awt.Dimension(579, 322));

        jPanel1.setBackground(new java.awt.Color(102, 0, 0));

        jLabel1.setFont(new java.awt.Font("Garamond", 1, 24)); //
NOI18N
        jLabel1.setForeground(new java.awt.Color(230, 230, 71));
        jLabel1.setText("Topic Label");

        jScrollPane1.setBackground(new java.awt.Color(255, 255,
255));
        jScrollPane1.getViewport().setBackground(new
java.awt.Color(255, 255, 255));
        jScrollPane1.setBorder(null);

jScrollPane1.setHorizontalScrollBarPolicy(javax.swing.ScrollPan
eConstants.HORIZONTAL_SCROLLBAR_NEVER);

        jTextArea1.setEditable(false);
        jTextArea1.setColumns(20);
        jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 14)); //
NOI18N
        jTextArea1.setLineWrap(true);
        jTextArea1.setRows(5);
        jTextArea1.setTabSize(5);
        jTextArea1.setWrapStyleWord(true);
        jScrollPane1.setViewportView(jTextArea1);

        jButton1.setText("Close Window");
        jButton1.addActionListener(new
java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent
evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.Grou
pLayout.Alignment.LEADING)
                    .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jScrollPane1))
                .addContainerGap())

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap(243, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addGap(237, 237, 237))
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 35,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 230,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                .addComponent(jButton1)
                .addContainerGap())
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING)
            .addComponent(jPanel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );

        pack();
    }// </editor-fold>//GEN-END:initComponents

    private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_jButton1ActionPerformed
        this.dispose();
    }//GEN-LAST:event_jButton1ActionPerformed

    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel
setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay
with the default look and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/pl
af.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(helpScreen.class.getName()).
```

```
log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(helpScreen.class.getName()).
log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(helpScreen.class.getName()).
log(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException
ex) {

java.util.logging.Logger.getLogger(helpScreen.class.getName()).
log(java.util.logging.Level.SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new helpScreen().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton jButton1;
    protected javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    protected javax.swing.JTextArea jTextArea1;
    // End of variables declaration//GEN-END:variables
}


package my.reversiblewatermarking;

import java.io.File;
import java.util.StringTokenizer;
import javax.swing.JOptionPane;

/**
 * @author Dan Pantano
 */
class Application_handler {
    public Congruence_embedder embedder;
    public Congruence_extractor extractor;
    private File[] carrierImages;
    private File watermark;
    private File[] inputImages;
    private String encryptionKey;
    private String decryptionKey;

    Application_handler(){
        embedder = new Congruence_embedder();
        extractor = new Congruence_extractor();
    }

    void set_carrierImages(File[] images){
        this.carrierImages = images;
    }

    void set_watermark(File watermark){
        this.watermark = watermark;
    }

    void set_inputImages(File[] images){
        this.inputImages = images;
    }

    void set_encryptionKey(String password){
        this.encryptionKey = password;
    }

    void set_decryptionKey(String password){
        this.decryptionKey = password;
    }

    File[] get_carrierImages(){
        return this.carrierImages;
    }

    File get_watermark(){
        return this.watermark;
    }

    File[] get_inputImages(){
        return this.inputImages;
    }

    String get_encryptionKey(){
        return this.encryptionKey;
    }

    String get_decryptionKey(){
        return this.decryptionKey;
    }

    boolean check_carrierImages(){
        if(this.carrierImages==null){
            return false;
        }
        else{
            return true;
        }
    }

    boolean check_watermark(){
        if(this.watermark==null){
            return false;
        }
        else{
            return true;
        }
    }

    boolean check_inputImages(){
        if(this.inputImages==null){
            return false;
        }
        else{
            return true;
        }
    }

    boolean check_encryptionKey(){
        if(this.encryptionKey==null){
            return false;
        }
        else{
            return true;
        }
    }

    boolean check_decryptionKey(){
        if(this.decryptionKey==null){
            return false;
        }
        else{
            return true;
        }
```

```
    }

    void clear_input(){
        this.carrierImages = null;
        this.watermark = null;
        this.inputImages = null;
        this.encryptionKey = null;
        this.decryptionKey = null;
    }

    void extract(resultScreen s){
        File[] images = this.get_inputImages();
        String result = "Success";
        String status = "";
        StringTokenizer tokenized;
        int counter = 1;

        if(images==null){
            JOptionPane.showMessageDialog(null, "Unable to start
watermark "
            + "extraction and image restoration due to incomplete
input.");
        }
        else{
            s.setVisible(true);
            s.update("Starting extraction");

            for(int i=0; i<images.length; i++){
                tokenized = new StringTokenizer(result);
                status = tokenized.nextToken();
                s.update("\nImage " + (i+1) + ": " +
images[i].getName()+ ":is now being "
                        + "restored and the watermark is being
extracted.");
                if(images.length==1){
                    //don't change the filename
                }
                else{
                    if(counter>1){
                        String fname =
this.extractor.get_restored_filename();
                        fname =
fname.replace(fname.substring(fname.length()-2,
fname.length()-1), Integer.toString(counter));
                        if(status.equalsIgnoreCase("Error")){
                            //don't change the filename
                        }
                        else{
                            this.extractor.set_restored_filename(fname);
                            counter++;
                        }
                    }else{
                        if(status.equalsIgnoreCase("Error")){
                            //don't change the filename
                        }
                        else{

this.extractor.set_restored_filename(this.extractor.get_restored_f
ilename()+"("+counter+")");
                            counter++;
                        }
                    }
                }
                result = this.extractor.extract(images[i],
this.decryptionKey);
                s.update("\t ->" + result);
            }
            this.set_inputImages(null);
        }
        s.exit_resultButton.setEnabled(true);
    }

    void embed(resultScreen s){
        File[] images = this.get_carrierImages();
        File watermark = this.get_watermark();
        String result = "Success";
        String status = "";
        StringTokenizer tokenized;
        int counter = 1;

        if(images==null || watermark==null){
            JOptionPane.showMessageDialog(null, "Unable to start
watermark "
            + "embedding due to incomplete input.");
        }
        else{
            s.setVisible(true);
            s.update("Starting watermark embedding");

            for(int i=0; i<images.length; i++){
                tokenized = new StringTokenizer(result);
                status = tokenized.nextToken();
                s.update("\nImage " + (i+1) + ": " +
images[i].getName()+ " is now being "
                        + "embedded with the watermark.");
                if(images.length==1){
                 //don't change the filename
                }
                else{
                    if(counter>1){
                        String fname = this.embedder.get_filename();
                        fname =
fname.replace(fname.substring(fname.length()-2,
fname.length()-1), Integer.toString(counter));
                        if(status.equalsIgnoreCase("Error")){
                            //don't change the filename
                        }
                        else{
                            this.embedder.set_filename(fname);
                            counter++;
                        }
                    }else{
                        if(status.equalsIgnoreCase("Error")){
                            //don't change the filename
                        }
                        else{

this.embedder.set_filename(this.embedder.get_filename()
+"("+counter+")");
                            counter++;
                        }
                    }
                }
                result = this.embedder.embed(images[i], watermark,
this.encryptionKey);
                s.update("\t ->" + result);
            }
            this.set_carrierImages(null);
            this.set_watermark(null);
        }
        s.exit_resultButton.setEnabled(true);
    }
}


package my.reversiblewatermarking;

import java.awt.image.BufferedImage;
import java.awt.image.Raster;
import java.awt.image.WritableRaster;
```

```
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.security.Security;
import java.util.ArrayList;
import javax.crypto.Cipher;
import javax.crypto.CipherOutputStream;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;

import javax.imageio.ImageIO;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class Congruence_embedder {
	private static String watermarked_name;
	private static String watermarked_path;
	public final String WATERMARKED_EXT = ".png";
	public final String WATERMARKED_DEF =
"watermarkedimage";

	public Congruence_embedder(){
	  this.watermarked_name = "watermarkedimage";
	  this.watermarked_path = System.getProperty("user.dir");
	}

	public static void set_filename(String filename){
	  watermarked_name = filename;
	}

	public static String get_filename(){
	  return watermarked_name;
	}

	public static void set_filepath(String filepath){
	  watermarked_path = filepath;
	}

	public static String get_filepath(){
	  return watermarked_path;
	}

	public String embed(File carrier, File watermark, String
encryptionkey){
		String invalidcarrier = "Error : Invalid type of
carrier image.";
		String invalidwatermark = "Error : Invalid
type of watermark.";
		String notenoughembedding = "Error : There
is not enough space to embed the watermark.";
		String carrier_ioerror = "Error : Carrier image
cannot be opened.";
	String carrier_typeerror = "Error : Carrier image cannot
be opened because of color type.";
		String watermarkfof = "Error : Watermark file
cannot be opened.";
		String embeddingerror = "Error : Embedding
process error.";
	String encrypterror = "Error : Encryption process
error.";
	String colorerror = "Error : Invalid type of color model.
(Image is not grayscale)";
		String success = "Success : " +
watermark.getName() + " has successfully been embedded into
" + this.get_filename() + this.WATERMARKED_EXT;
	String success_enc = "Success : " +
watermark.getName() + " has successfully been embedded and
encrypted into " + this.get_filename() +
this.WATERMARKED_EXT;

		String carrier_name = carrier.getName();
		String watermark_name =
watermark.getName();
		String carrier_ext =
carrier_name.substring(carrier_name.length() - 3);
		String watermark_ext =
watermark_name.substring(watermark_name.length() - 3);
		String result_name = this.get_filepath() + "\\"
+ this.get_filename() + this.WATERMARKED_EXT;
		int carrier_type = get_filetype(carrier_ext);
		int watermark_type =
get_filetype(watermark_ext);

		//check for file type validity
		if(carrier_type==4 || carrier_type==0){
			return invalidcarrier;
		}
		if(watermark_type==0){
			return invalidwatermark;
		}

		BufferedImage carrier_image;
		try{
			carrier_image =
ImageIO.read(carrier);
		}
		catch(Exception e){
			return carrier_ioerror;
		}

		Raster carrier_raster =
carrier_image.getData();
	//list of pixels
		int[] pixelList = new
int[carrier_raster.getWidth() * carrier_raster.getHeight()];
		//get pixel by pixel
		int pix = 0;

		//pixel counter
		int ctr = 0;

	int rgb=0, r=0, g=0, bl=0;
	//get the image in the array
		for(int i = 0 ; i < carrier_raster.getWidth() ; i+
+){
		  for(int j = 0 ; j < carrier_raster.getHeight() ; j++)
			{
				try{
			  rgb = carrier_image.getRGB(i, j);
			  r = (0x00ff0000 & rgb) >> 16;
			  g = (0x0000ff00 & rgb) >> 8;
			  bl = (0x000000ff & rgb);
			  if(r!=g || r!=bl || g!=bl){
			    return colorerror;
			  }
			  pix = carrier_raster.getSample(i, j, 0);
			  pixelList[ctr] = pix;
			  ctr++;
			}
			catch(Exception e){
			  return carrier_typeerror;
			}
				}
			}

		ArrayList<Integer> embeddingList = new
ArrayList<Integer>();
		ArrayList<Integer> tocorrectList = new
```

```
ArrayList<Integer>();

                        int transformtest = 0;
                        int prevstorage = 0;
                        int next = 0;
                        boolean tocorrect = false;
                        int k = 0;
                        int interval = 0;
                        int corrective = 0;
                        int addcorrective = 0;
            int steps = 0;
            int total = carrier_raster.getWidth() *
carrier_raster.getHeight();

            for(int i = 0; i < (pixelList.length-1); i++){
               if(steps>total){
                  return embeddingerror;
               }
               transformtest = Ttransform(pixelList[i],
pixelList[i+1]);
               steps++;

               //check for embedding pixel
               if((0 <= transformtest && (transformtest+4) <= 255))
{
                  prevstorage = pixelList[i];
                  pixelList[i] = transformtest;
                  embeddingList.add(i);
               }
               else{
                  next = next_embedding(i, pixelList);
                  if(next == -1){
                     break;
                  }

                  interval = count_interval(i, next);

                  if(interval%2!=0 && i==0){

                  }

                  if(interval%2!=0){
                     k = i - 1;
                     pixelList[(i-1)] = prevstorage;

embeddingList.remove(embeddingList.indexOf((i-1)));
                  }
                  else{
                     k = i;
                  }

                  for(int j=k; j<next; j++){
                     if(tocorrect){
                        corrective = get_corrective(j, pixelList);
                        pixelList[j] = pixelList[j] - corrective;

                        if((corrective>=0 && corrective<=1)){
                              tocorrectList.add((corrective+1));
                        }
                        else if((corrective>=2 && corrective<=4)){
                              addcorrective = corrective - 4 + 3;
                              corrective = 3;
                              tocorrectList.add(corrective);
                              tocorrectList.add(addcorrective);

                        }
                        else if((corrective<=-1 && corrective>=-4)){
                              addcorrective = -1 * corrective;
                              corrective = 4;
                              tocorrectList.add(corrective);
                              tocorrectList.add(addcorrective);
                        }
                        else{
                              return embeddingerror;
                        }
                     }
                     else{
                        //original pixel, not to be modified
                     }
                     tocorrect = !tocorrect;
                  }
                  tocorrect = false;
                  i = next - 1;
               }
                        }

                        byte[] watermarkBytes;

            if(watermark_type==4){
               //text file watermark
               FileInputStream fis;
               try{
                     fis = new FileInputStream(watermark);
               }
               catch(Exception e){
                     return watermarkfof;
               }
               ByteArrayOutputStream bos = new
ByteArrayOutputStream();
               byte[] buf = new byte[1024];
               try{
                     for(int readNum; (readNum = fis.read(buf)) !=
-1;){
                           bos.write(buf, 0, readNum); //no doubt
here is 0
                  //Writes len bytes from the specified byte array
starting at offset off to this byte array output stream.
                        }
                        bos.flush();
                        watermarkBytes = bos.toByteArray();
                        bos.close();
               }
               catch(Exception ex){
                     return embeddingerror;
               }
                        }
                        else{
               //image watermark
               if(watermark_type==2){
                  BufferedImage originalImage;
                  try {
                        originalImage = ImageIO.read(watermark);
                        ByteArrayOutputStream baos = new
ByteArrayOutputStream();
                        ImageIO.write(originalImage,
watermark_ext, baos);
                        // convert BufferedImage to byte array
                        baos.flush();
                        watermarkBytes = baos.toByteArray();
                        baos.close();
                  } catch (Exception e) {
                        return embeddingerror;
                  }
               }
               else{
                  BufferedImage originalImage;
                  try {
                        originalImage = ImageIO.read(watermark);
                        ByteArrayOutputStream baos = new
ByteArrayOutputStream();
```

```
                    ImageIO.write(originalImage,
watermark_ext, baos);
                    // convert BufferedImage to byte array
                    baos.flush();
                    watermarkBytes = baos.toByteArray();
                    baos.close();
                } catch (Exception e) {
                    return embeddingerror;
                }
            }
                    }

                    ctr=0;
                    StringBuilder binary = new StringBuilder();
                    for(byte b : watermarkBytes){
            int val = b;
            for(int i = 0; i < 8; i++){
                binary.append((val & 128) == 0 ? 0 : 1);
                val <<= 1;
                ctr++;
            }
                    }

                    //composition of the message to be
embedded
                    String switcher = "";
                    ArrayList<Integer> embedmessage = new
ArrayList();

                    //embed watermark file type
                    embedmessage.add(watermark_type);

                    String binarized = Integer.toBinaryString(ctr);
                    int len = binarized.length();
                    String zeroes =
"00000000000000000000000000000000000000000000000000000000
0000000000000";
                    if (len < 64)
                            binarized = zeroes.substring(0, 64-
len).concat(binarized);
                    else
                        binarized = binarized.substring(len - 64);

                    for(int i=0; i<binarized.length(); i++){
            switcher = Character.toString(binarized.charAt(i)) +
Character.toString(binarized.charAt((i+1)));
            if(get_toembed(switcher)==0){
                return embeddingerror;
            }
            embedmessage.add(get_toembed(switcher));
            i = i + 1;
                    }

                    for(int i=0; i<binary.length(); i++){
            switcher = Character.toString(binary.charAt(i)) +
Character.toString(binary.charAt((i+1)));
            if(get_toembed(switcher)==0){
                return embeddingerror;
            }
            embedmessage.add(get_toembed(switcher));
            i = i + 1;
                    }

                    binarized =
Integer.toBinaryString(tocorrectList.size());
                    len = binarized.length();
                    if(len < 64)
            binarized = zeroes.substring(0, 64-
len).concat(binarized);
                    else
                        binarized = binarized.substring(len - 64);

                    for(int i=0; i<binarized.length(); i++){
            switcher = Character.toString(binarized.charAt(i)) +
Character.toString(binarized.charAt((i+1)));
            if(get_toembed(switcher)==0){
                return embeddingerror;
            }
            embedmessage.add(get_toembed(switcher));
            i = i + 1;
                    }

                    for(int i=0; i<tocorrectList.size(); i++){
            embedmessage.add(tocorrectList.get(i));
                    }

                    embedmessage.add(carrier_type);

                    int totalnumber = tocorrectList.size() +
embedmessage.size();
                    // end embed composition

                    //start embedding
                    int embedded = 0;
                    if(totalnumber>embeddingList.size()){
            return notenoughembedding;
                    }
                    else{
            for(int i=0; i<embeddingList.size(); i++){
                if(i<embedmessage.size()){
                    if(embedmessage.get(i)==0){
                        return embeddingerror;
                    }
                    pixelList[embeddingList.get(i)] +=
embedmessage.get(i);
                }
                else{
                    pixelList[embeddingList.get(i)] += 4;
                }
                embedded++;
            }

            BufferedImage processedImage = new
BufferedImage(carrier_raster.getWidth(),
carrier_raster.getHeight(),BufferedImage.TYPE_BYTE_GRAY);
            WritableRaster raster =
processedImage.getRaster();
            ctr = 0;
            for(int i = 0; i<carrier_raster.getWidth(); i++){
                for(int j = 0; j<carrier_raster.getHeight();j++){
                    raster.setSample(i,j,0,pixelList[ctr]);
                    ctr++;
                }
            }

            try {
                ImageIO.write(processedImage, "PNG", new
File(result_name));
            } catch (Exception e) {
                return embeddingerror;
            }
                    }

        if(encryptionkey!=null){
            Security.insertProviderAt(new
BouncyCastleProvider(), 1);
            byte[] salt = { (byte) 0xc7, (byte) 0x73, (byte) 0x21,
(byte) 0x8c,
            (byte) 0x7e, (byte) 0xc8, (byte) 0xee, (byte) 0x99 };
            File inputFile = new File(result_name);
```

```
            BufferedImage input = null;
            try {
               input = ImageIO.read(inputFile);
            } catch (Exception ex) {
               return encrypterror;
            }

            SecretKeyFactory keyFac = null;
            try {
               keyFac =
SecretKeyFactory.getInstance("PBEWITHSHA256AND128BITA
ES-CBC-BC");
            } catch (Exception ex) {
               return encrypterror;
            }

            PBEKeySpec pbeKeySpec = new
PBEKeySpec(encryptionkey.toCharArray());
            PBEParameterSpec pbeParamSpec = new
PBEParameterSpec(salt, 20);

            SecretKey pbeKey = null;
            try {
               pbeKey = keyFac.generateSecret(pbeKeySpec);
            } catch (Exception ex) {
               return encrypterror;
            }

            // Make a PBE Cyhper object and initialize it to
encrypt using
            // the given password.
            Cipher pbeCipher = null;
            try {
               pbeCipher =
Cipher.getInstance("PBEWITHSHA256AND128BITAES-CBC-
BC");
            } catch (Exception ex) {
               return encrypterror;
            }

            try {
               pbeCipher.init(Cipher.ENCRYPT_MODE,
pbeKey, pbeParamSpec);
            } catch (Exception ex) {
               return ex.getMessage();
            }


            FileOutputStream output = null;
            try {
               output = new FileOutputStream(result_name);
            } catch (Exception ex) {
               return encrypterror;
            }

            CipherOutputStream cos = new
CipherOutputStream(output, pbeCipher);

            try {
               ImageIO.write(input,"PNG",cos);
            } catch (Exception ex) {
               return encrypterror;
            }

            try {
               cos.close();
            } catch (Exception ex) {
               return encrypterror;
            }
```

```
         return success_enc;
      }

            return success;
      }

      private static int next_embedding(int start, int[] pixels){
            int t_temp = 0;
            for(int i = start; i < pixels.length; i++){
                  t_temp = Ttransform(pixels[i],
pixels[i+1]);
                   if(0 <= t_temp && (t_temp+4) <=
255){
                        return i;
                  }
                  else{
                  }
            }
            return -1;
      }

      private static int Ttransform(int x1, int x2){
            int result = ((4+1) * x1) - (4 * x2);
            return result;
      }

      private static int count_interval(int start, int end){
            int result = 0;
            for(int i = start; i < end; i++){
                  result++;
            }
            return result;
      }

      private static int get_corrective(int pos, int[] pixels){
            int c = 0;
            c = (pixels[pos] + (4*pixels[pos+1])) % (4+1);
            if((pixels[pos]-c)<0){
                  c = (4 + 1 - c) * -1;
            }
            return c;
      }

      private static int get_filetype(String ext){
            if(ext.equalsIgnoreCase("jpg")){
                  return 1;
            }
            else if(ext.equalsIgnoreCase("bmp")){
                  return 2;
            }
            else if(ext.equalsIgnoreCase("png")){
                  return 3;
            }
            else if(ext.equalsIgnoreCase("txt")){
                  return 4;
            }
            else{
        return 0;
            }
      }

      private static int get_toembed(String checker){
            if(checker.equalsIgnoreCase("00")){
                  return 1;
            }
            else if(checker.equalsIgnoreCase("01")){
                  return 2;
            }
            else if(checker.equalsIgnoreCase("10")){
                  return 3;
```

```
                        }
                        else if(checker.equalsIgnoreCase("11")){
                                return 4;
                        }
                        else{
                                return 0;
                        }
                }
}


package my.reversiblewatermarking;

import java.awt.image.BufferedImage;
import java.awt.image.Raster;
import java.awt.image.WritableRaster;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.math.BigInteger;
import java.security.Security;
import java.util.ArrayList;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;

import javax.imageio.ImageIO;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class Congruence_extractor {
            private static int n= 4;
            private static String restored_name;
            private static String watermark_name;
        private static String save_filepath;
        public final String RESTORED_DEF = "restoredimage";
        public final String WATERMARK_DEF = "watermark";

            public Congruence_extractor(){
           restored_name = "restoredimage";
           watermark_name = "watermark";
           this.save_filepath = System.getProperty("user.dir");
            }

            public static String get_restored_filename(){
           return restored_name;
            }

            public static String get_watermark_filename(){
           return watermark_name;
            }

        public void set_restored_filename(String filename){
            this.restored_name = filename;
        }

        public void set_watermark_filename(String filename){
            this.watermark_name = filename;
        }

        public void set_filepath(String filepath){
            this.save_filepath = filepath;
        }

            public String get_filepath(){
           return this.save_filepath;
            }

        }

        public String extract(File carrier, String decryptionkey){
                String invalidcarrier = "Error : Invalid type of
watermarked image.";
                String watermarkfof = "Error : Watermarked
image cannot be opened.";
            String colorerror = "Error : Invalid type of color model.
(Image is not grayscale)";
                String extracterror = "Error : Watermark
cannot be extracted.(Watermarked image error)";
            String decrypterror = "Error : Decryption Process
Error";
                String success = "Success : " +
this.get_restored_filename();

                String carrier_name = carrier.getName();
                String carrier_ext =
carrier_name.substring(carrier_name.length() - 3);

                String restore_name = this.get_filepath() +
"\\" + this.get_restored_filename();
                String extracted_name_path =
this.get_filepath() + "\\" + this.get_watermark_filename();
            String extracted_name =
this.get_watermark_filename();

                int carrier_type = get_filetype(carrier_ext);

                if(carrier_type!=3){
                        return invalidcarrier;
                }

            if(decryptionkey!=null){
               Security.insertProviderAt(new
BouncyCastleProvider(), 1);
               byte[] salt = { (byte) 0xc7, (byte) 0x73, (byte) 0x21,
(byte) 0x8c,
               (byte) 0x7e, (byte) 0xc8, (byte) 0xee, (byte) 0x99 };

               File inputFile = carrier, backup = carrier;
               BufferedImage input;
               try {
                  input = ImageIO.read(inputFile);
               } catch (Exception ex) {
                  return decrypterror;
               }

               PBEKeySpec pbeKeySpec = new
PBEKeySpec(decryptionkey.toCharArray());

               PBEParameterSpec pbeParamSpec = new
PBEParameterSpec(salt, 20);
               SecretKeyFactory keyFac;
               try {
                  keyFac =
SecretKeyFactory.getInstance("PBEWITHSHA256AND128BITA
ES-CBC-BC");
               } catch (Exception ex) {
                  return decrypterror;
               }

               SecretKey pbeKey;
               try {
                  pbeKey = keyFac.generateSecret(pbeKeySpec);
               } catch (Exception ex) {
                  return decrypterror;
               }

               Cipher pbeCipher;
```

```
            try {
                pbeCipher =
Cipher.getInstance("PBEWITHSHA256AND128BITAES-CBC-
BC");
            } catch (Exception ex) {
                return decrypterror;
            }

            try {
                pbeCipher.init(Cipher.DECRYPT_MODE,
pbeKey, pbeParamSpec);
            } catch (Exception ex) {
                return decrypterror;
            }

            FileInputStream fis;
            try {
                fis = new FileInputStream(inputFile);
            } catch (Exception ex) {
                return decrypterror;
            }

            CipherInputStream cis = new CipherInputStream(fis,
pbeCipher);
            try {
                // We then use all that to read the image
                input = ImageIO.read(cis);
            } catch (Exception ex) {
                return decrypterror;
            }

            try {
                cis.close();
            } catch (Exception ex) {
                return decrypterror;
            }

            FileOutputStream output;
            String decrypted_filename =
carrier.getAbsolutePath();
            try {
                output = new
FileOutputStream(decrypted_filename);
            } catch (Exception ex) {
                return decrypterror;
            }

            try {
                ImageIO.write(input, "PNG", output);
                carrier = new File(decrypted_filename);
            } catch (Exception ex) {
                return decrypterror;
            }
        }

                BufferedImage image;
        Raster image_raster;
                try {
                        image = ImageIO.read(carrier);
            image_raster = image.getData();
        } catch (Exception e) {
                        return watermarkfof;
                }

                //list of pixels
                int[] pixelList = new
int[image_raster.getWidth() * image_raster.getHeight()];
                int[] pixelList2 = new
int[image_raster.getWidth() * image_raster.getHeight()];

                //get pixel by pixel
                int pix = 0;

                int ctr = 0;

        int rgb=0, r=0, g=0, bl=0;

                //get the image in the array
                for(int i = 0 ; i < image_raster.getWidth() ; i+
+){
                        for(int j = 0 ; j <
image_raster.getHeight() ; j++){
                try{
                    rgb = image.getRGB(i, j);
                    r = (0x00ff0000 & rgb) >> 16;
                    g = (0x0000ff00 & rgb) >> 8;
                    bl = (0x000000ff & rgb);
                    if(r!=g || r!=bl || g!=bl){
                        return colorerror;
                    }
                    pix = image_raster.getSample(i, j, 0);
                    pixelList[ctr] = pix;
                    pixelList2[ctr] = pix;
                    ctr++;
                }
                catch(Exception e){
                    return e.toString();
                }
                        }
                }

                int v = 0;

                ArrayList<Integer> extractList = new
ArrayList<Integer>();
                ArrayList<Integer> tocorrectList = new
ArrayList<Integer>();
                ArrayList<String> watermarkList = new
ArrayList<String>();

                for(int i=(pixelList.length-2); i>=0; i--){
                        v = (pixelList[i]+4*pixelList2[(i+1)])
% (4 + 1);

                        if(v==0){
                                tocorrectList.add(i);
                                i -= 1;
                        }
                        else{
                                extractList.add(v);
                                pixelList[i] -= v;
                                pixelList2[i] = (pixelList[i]
+4*pixelList2[(i+1)]) / (4 + 1);
                        }
                }

                for(int i=(extractList.size()-1); i>=0; i--){
                        v = extractList.get(i);

        if(get_tobits(v).equalsIgnoreCase("x")){
                                return extracterror;
                        }
                        watermarkList.add(get_tobits(v));
                }

                String watermark_ftype = "";
                String watermarktotal = "";
                String tocorrecttotal = "";
                String watermark_bits = "";
                int pos = 0;
```

```
                        //get watermark file type
                        if(watermarkList.size()==0){
                                return extracterror;
                        }
                        watermark_ftype =
get_filetype2(watermarkList.get(0));
                        extracted_name += watermark_ftype;
                extracted_name_path += watermark_ftype;
                        String watermark_ext =
watermark_ftype.substring(watermark_ftype.length()-
3).toUpperCase();
                        int watermark_embedded =
get_toembed(watermarkList.get(0));
                        if(watermark_embedded==0){
                                return extracterror;
                        }
                        pos++;

                        //get total of watermark bits
                        for(int i=1; i<=32; i++){
                                watermarktotal +=
watermarkList.get(i);
                                pos++;
                        }
                int base = 2;

                int decimal = 0;
                try{
                   decimal = Integer.parseInt(watermarktotal, base);
                }catch(Exception e){
                   return extracterror;
                }
                int watermark_limit = decimal/2;

                        //get watermark bits
                        int realpos = 0;
                        for(int i=0; i<watermark_limit; i++){
                                realpos = 33 + i;
                                watermark_bits +=
watermarkList.get(realpos);
                                pos++;
                        }

                        //get total of corrective code bits
                        for(int i=0; i<32; i++){
                                realpos = (watermark_limit+33) + i;
                                tocorrecttotal +=
watermarkList.get(realpos);
                                pos++;
                        }

                        int corrective_limit =
Integer.parseInt(tocorrecttotal, base);

                        ArrayList<Integer> precorrectiveCodes =
new ArrayList<Integer>();
                        ArrayList<Integer> correctiveCodes = new
ArrayList<Integer>();

                        int ccode = 0;

                        //get pre corrective codes
                        for(int i=pos; i<(pos+corrective_limit); i++){
                                ccode =
get_toembed(watermarkList.get(i));
                                if(ccode>4 || ccode<=0 ){
                                        return extracterror;
                                }
                                precorrectiveCodes.add(ccode);
                        }
                        pos += corrective_limit;
                        String origtype =
get_filetype2(watermarkList.get(pos));
                        int orig_type =
get_toembed(watermarkList.get(pos));
                        if(orig_type==0 || orig_type==4){
                                return extracterror;
                        }
                        restore_name += origtype;
                        String orig_ext =
origtype.substring(origtype.length()-3).toUpperCase();
                        success += origtype;
                success += " has successfully been restored.";

                        ccode = 0;
                        int w2 = 0;
                        for(int i=0; i<precorrectiveCodes.size();i++){
                                ccode = precorrectiveCodes.get(i);

                                if(ccode==1 || ccode==2){
                                        ccode = ccode - 1;
                                }
                                else if(ccode==3){
                                        w2 =
precorrectiveCodes.get((i+1));
                                        ccode = w2 + 4 - 3;
                                        i += 1;
                                }
                                else if(ccode==4){
                                        w2 =
precorrectiveCodes.get((i+1));
                                        ccode = -1 * w2;
                                        i += 1;
                                }
                                else{
                                        return extracterror;
                                }

                                if(ccode>4 || ccode <-4){
                                        return extracterror;
                                }
                                correctiveCodes.add(ccode);
                        }

                        int tocorrect_limit = tocorrectList.size();
                        for(int i=0; i<tocorrectList.size(); i++){
                try{
                                        tocorrect_limit--;
                                        pixelList2[tocorrectList.get(i)] =
pixelList[tocorrectList.get(i)] +
correctiveCodes.get(tocorrect_limit);
                        }catch(Exception ex){
                           return extracterror;
                        }
                        }

                        //restore the original image
                        BufferedImage processedImage = new
BufferedImage(image_raster.getWidth(),
image_raster.getHeight(),BufferedImage.TYPE_BYTE_GRAY);
                        WritableRaster raster =
processedImage.getRaster();
                        ctr = 0;
                        for(int i = 0; i<image_raster.getWidth(); i++){
                                for(int j = 0;
j<image_raster.getHeight();j++){

                raster.setSample(i,j,0,pixelList2[ctr]);
                                        ctr++;
```

```
                        }
                    }
                    try {
                        ImageIO.write(processedImage,
"PNG", new File(restore_name));
                    } catch (Exception e) {
                        return extracterror;
                    }

            byte[] bytes;
                    bytes = new BigInteger(watermark_bits,
2).toByteArray();
                    byte[] result = new byte[(bytes.length-1)];
                    for(int i=1; i<bytes.length; i++){
                        result[(i-1)] = bytes[i];
                    }

                    if(watermark_embedded==4){
                        File someFile = new
File(extracted_name_path);
                        FileOutputStream fos;
                        try {
                            fos = new
FileOutputStream(someFile);
                            fos.write(bytes);
                            fos.flush();
                            fos.close();
                        } catch (Exception e) {
                            return extracterror;
                        }
                    }
                    else{
                if(watermark_embedded==2){
                    BufferedImage bImageFromConvert;
                    try {
                        bImageFromConvert = ImageIO.read(new
ByteArrayInputStream(bytes));
                        ImageIO.write(bImageFromConvert,
"BMP", new File(extracted_name_path));
                    } catch (Exception e) {
                        return e.getMessage();
                    }
                }
                else{
                    InputStream in = new
ByteArrayInputStream(result);
                    BufferedImage bImageFromConvert;
                    try {
                        bImageFromConvert = ImageIO.read(in);
                        ImageIO.write(bImageFromConvert,
watermark_ext, new File(extracted_name_path));
                    } catch (Exception e) {
                        return e.getMessage();
                    }
                }
                    }

                    success += " " + extracted_name + " was the
watermark extracted.";
                    return success;
            }

            private static int get_filetype(String ext){
                    if(ext.equalsIgnoreCase("jpg")){
                        return 1;
                    }
                    else if(ext.equalsIgnoreCase("bmp")){
                        return 2;
                    }
                    else if(ext.equalsIgnoreCase("png")){
                        return 3;
                    }
                    else if(ext.equalsIgnoreCase("txt")){
                        return 4;
                    }
                    else{
                        return 0;
                    }
            }

            private static String get_tobits(Integer checker){
                    if(checker==1){
                        return "00";
                    }
                    else if(checker==2){
                        return "01";
                    }
                    else if(checker==3){
                        return "10";
                    }
                    else if(checker==4){
                        return "11";
                    }
                    else{
                        return "x";
                    }
            }

            private static String get_filetype2(String checker){
                    if(checker.equalsIgnoreCase("00")){
                        return ".jpg";
                    }
                    else if(checker.equalsIgnoreCase("01")){
                        return ".bmp";
                    }
                    else if(checker.equalsIgnoreCase("10")){
                        return ".png";
                    }
                    else if(checker.equalsIgnoreCase("11")){
                        return ".txt";
                    }
                    else{
                        return "";
                    }
            }

            private static int get_toembed(String checker){
                    if(checker.equalsIgnoreCase("00")){
                        return 1;
                    }
                    else if(checker.equalsIgnoreCase("01")){
                        return 2;
                    }
                    else if(checker.equalsIgnoreCase("10")){
                        return 3;
                    }
                    else if(checker.equalsIgnoreCase("11")){
                        return 4;
                    }
                    else{
                        return 0;
                    }
            }
}
```

## XI. Acknowledgement

The journey was long and arduous. It had it's up and downs, and the completion of this special problem is like a sign that says this journey will be ending soon. But success would not have been possible without the people who shared this journey with me. Mere words do not do justice to how grateful I am to these people.

Above all, I would like to thank my mother for her unwavering support through everything I have gone through. She is always there for me, even though she has her own share of problems to deal with. My sisters, Deeva, Desi, Danae and Danna have given me support throughout as always, for which a mere 'thank you' will not suffice.

I also express my sincerest gratitude to my adviser, Dr. Vincent Peter Magboo. Despite his busy schedule, he still found the time to guide me through my proposal and my defense. I learned a lot from him, and his guidance was extremely valuable to the completion of this special problem.

It gives me pleasure to acknowledge Ms. Rose Ann Organo for the inspiration she provided in making this special problem.  I hope that she never forgets me, and when the dust settles, I hope she sees that I'll always be there for her.

I would also like to thank my Computer Science professors and instructors whose lessons were greatly beneficial to the development of my application.

I am also grateful to Dr. Villarta and the people of the IMS. They were very patient and understanding while I was working on this special problem. Their support and encouragement is very much appreciated.

Last, but by no means least, I thank all the wonderful and extremely awesome men and women of ComSci Dose '08. College life was difficult and exhausting, but you guys made it fun. I'll never forget all the parties, the games, the laughs, jokes and even the arguments. I feel extremely lucky to be a part of this GREAT block. I see them not just friends but also as family.  I might not have mentioned them all one by one but they have all helped me become who I am today.

From personal experience, I have seen that effort and perseverance does not assure success. But this time, I'm glad it did. I thank everyone who has been with me throughout this magnificent journey.