Software Requirements Specification

For

Paranoia

U-S-E
CS169 – Software Engineering
Sept. 30, 2003

Prepared by:

BERDAHL, EDGAR
CHUNG, SANG
GONG, GARY
KRUGLIKOVA, TATYANA
LE, TAM
LEE, JOON YUL
LI, SHENG
MOZAFFARIAN, BEHRAD
MURRAY, JASON
NGUYEN, HIEP
SHKOLNIKOV, YURIY

Table of Contents

1. INTRODUCTION

1.1 ABSTRACT

Paranoia allows an average user to securely transfer text messages by hiding them in a digital image file. A combination of steganography and encryption algorithms provides a strong backbone to *Paranoia's* security. *Paranoia* features innovative techniques for using the digital image file by hiding text in it or even using it as a key for the encryption.

1.2 RATIONALE

This project is feasible for several reasons. First, there is a lot of research that has been done in the field of steganography and encryption, so we will be able to reference pre-existing models. Also, Java has an extensive API that will serve the purpose of this project. We will use Java to program the user interface, which will greatly simplify creation of an easy-to-use GUI. Java is a good choice also because it has built-in capabilities for encryption and decryption.

This program would be valuable and innovative because of its intuitive interface and the format of the key as an image. Similar applications exist, but none of them use images as keys. This feature makes our application distinct because it adds an extra level of security. The idea of looking for a key in an image is not very obvious, and the hacker would need to have both images and *Paranoia* in order to easily check whether there is any hidden text. The user will only need to select the input files without any knowledge of underlying technologies.

The project contains several challenges that make it interesting to develop. The central task is to research available steganography and encryption algorithms to pick the ones that offer the best combination of strong encryption, usability, and performance as it applies to *Paranoia*. Another important aspect of the assignment is to create a simple, compact, and powerful GUI because it will play a large role in the success of the application. The GUI will include a walkthrough, which will be similar to a wizard to help first-time users. The user will also have a number of options – for example, he will be able to turn encryption on or off. The program window will be informative in general, since it will include image previews and a text box. The program should also gracefully handle errors, giving the user meaningful error messages. Another important goal is to make sure that the changes in the image are unnoticeable to human eyes and largely undetectable to programs designed to look for repetitive and obvious patterns.

1.3 INTENDED AUDIENCE

The application is primarily intended to be used to inconspicuously hide confidential and proprietary information by anyone seeking to hide information. This software has an advantage over other information security systems because both the key and the hidden text are in the form of images, which are not obvious text information carriers.

Because of its user-friendly interface, the application can also be used by anyone who wants to securely transmit private information. The main advantage of this program for individuals is that they do not have to have any knowledge about steganography or encryption. The visual way to encode the text, plus the visual key makes it easy for average users to navigate within the program.

1.4 SYSTEM REQUIREMENTS

Paranoia will be developed using Java Development Kit (JDK) 1.4.2. Any device that can support a Java Virtual Machine for JDK 1.4, along with the necessary memory and disk storage space will be able to execute our application.

1.5 SECURITY ASPECTS

Paranoia deals with secure communication between two parties – mainly hiding and recovering text that are embedded within a digital image file. *Paranoia* takes three steps to hide the data such that it is virtually impossible for a third party without a key to view or edit the information within the image.

1. Steganography (Steganography algorithm)

Our main tool of hiding text is steganography. Steganography is defined as the hiding of a secret message within another message in such a way that others do not discern the presence or contents of the hidden message - in our case, we hide a text message into a larger digital image file. This is achieved by hiding the text into the least significant bits (LSBs) of the larger image file. Because an image file whose LSBs have been changed still retains its functionality and is visibly similar to the original, steganography does a good job of hiding the data over an unconventional format. If no one is looking for the data, it's very hard to notice it.

2. Security through obscurity (Bit-placement algorithm)

We add another level of security by placing the sequential bits of the steganography algorithm in a random order. For example, if we had 4 bits to hide and there was a space of 10 bits available on the image file, we would not hide our 4 bits in bits 0 1 2 3, but rather create a pseudorandom bit placement such as bits 9 3 0 7 and use this for the first four bits.

3. Encryption (Encryption algorithm)

The last level and strongest level of security comes from encryption. Before we input our text message into the algorithms above, we encrypt it into ciphertext first. *Paranoia* will follow

an encryption algorithm given by the Advanced Encryption Standard (AES) to guarantee that our encryption is strong, reliable, and unbreakable through practical means.

Through the use of the three security algorithms above *Paranoia* will secure the communication between the two parties. In addition, *Paranoia* is based on using unconventional formats for keys and the medium of communication, and hence another level of security is added in that most people will neither suspect an incoming secret message embedded in an image nor a key generated from an image. Conclusively, *Paranoia* will be a secure tool for communication as the three algorithms in conjunction would be very hard if not impossible for a keyless third party to break.

2. DESCRIPTION OF MAIN FUNCTIONALITY

Paranoia is a platform-independent stand-alone application that combines steganography and encryption to enhance the confidentiality of intended message. The user's intended message is first encrypted to create unintelligible cipher text. Then the cipher text will be hidden within an image file in such a way as to minimize the perceived loss in quality. The recipient of the image is able to retrieve the hidden message back from the image with *Paranoia.*

2.1 ENCODING

In order to hide text in an image, the user must provide the text and a Target Image in which it is to be hidden. Optionally, users may enter or load a text key or generate a key from an image file. When a user key is not provided, a default key is used.

To make choosing images easier, an image bar is provided for the user. The user may set the source directory, whose contents are displayed as thumbnails in the image bar located on the bottom panel of application. These thumbnails are automatically generated from the specified source directory. The user then selects a target image and loads it for use as the Key Image or the Target Image. The user may also open the Target Image by selecting Open Image under File menu. Additionally, the Key Image may be opened with the Open Image Key command on the Tools menu.

The user may enter the text to be embedded in several ways. He or she may type it directly into the text window, open a text file, or paste text from another application. The user may also open a text file by selecting Open Text under the File menu or use the Edit menu to paste from the clipboard.

Loading a text key can be done from the Open Text Key selection on the Tools menu.

When user has specified both the target image, and text body, *Paranoia* is ready to hide the text body within the image. The user may then select Embed Text from the Tools menu. The key image will then be hashed into an appropriate Key Value. This value will be used for the encryption of the users plaintext to produce ciphertext. The ciphertext and key value will then be input into the steganography and bit placement algorithms, and the Output Image will be generated. After the application has generated the image, the user may then inspect it by selecting the Image tab on the main panel. The user is then able to compare the encrypted target image with the original target image located on the right box of the main window. The user may wish to save the generated image by selecting Save Image or Save Image As… under File menu.

2.2 DECODING

In order to retrieve the hidden text from a source image, the user needs to provide a Source Image and any Keys used when the Source Image was generated. Loading the Source Image file is done similarly to the process of opening a Target Image. If the sender has used the default key, the recipient need not load any keys to the application.

If a non-default key was used in text hiding process, the receiving party must have pre-arranged knowledge of the key for use in retrieving the text. This key could be an image, text file or a key string. If the agreed upon key was an image, the user can load it from the image browser into the key image box on the right. If the key is a text file, the user may load the key by selecting Open Text Key from the Tools menu. Lastly, the user may simply type in the key by activating the Text tab of the key panel and entering it into the text box there.

After the Source Image and any required Keys are loaded into the application, the hidden text can be retrieved by selecting Extract Text from the Tools menu. The key image will be hashed into an appropriate Key Value. This key value will be used to recover the hidden ciphertext from the Source Image. The key will then be used for the decryption of the ciphertext to produce plaintext. The plaintext will finally be displayed in the text box on the Text tab of the main panel. After the application has generated the text, the user may wish to save the generated text by selecting Save Text under the File menu.

The application is not able to identify the presence of the hidden message in images until the actual decoding process is attempted, as the image files are virtually indistinguishable from normal images.

2.3 THE USER WALKTHROUGH

A portion of the main GUI is devoted to the user walkthrough. This walkthrough will lead first time users through the process of encoding and decoding images and in its final step initiate the actual encoding or decoding of the user's text. The user may prepare for and perform encoding and decoding at any time without the use of the walkthrough.

Consult the following flow chart and corresponding description of states to understand the walkthrough process.

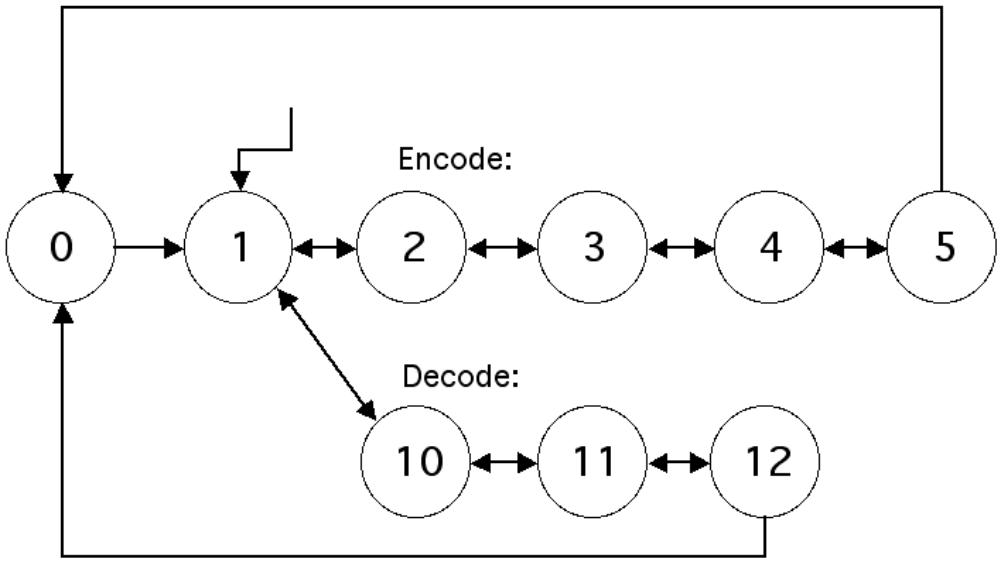


Figure 1. Flowchart for the walkthrough

The box below explains how each state will be described. Each has a unique state number and three buttons, some of which may be grayed out from time to time.

State n:	$Button_1$ (f_1)	$Button_2$ (f_2)	$Button_3$ (f_3)

n is the index of the state being described. $Button_1$, $Button_2$, and $Button_3$ are the names of the top, middle, and bottom buttons, respectively. f_1, f_2, and f_3 describe the following states if the button is clicked, respectively. If the value is gray, then the button should be grayed out and thus unclickable. Text similar to the text in this area should show up on the right side of the walkthrough in a text box.

```
Text in the Courier font consists of general notes
about the state.  Users should not be able to see
this text in the real walkthrough slides.
```

The following boxes detail each state.

State 1:	Embed (2)	Extract (10)	Cancel (gray)

This walkthrough can help you learn how to use Paranoia. Please select a process: *Embed* means hide text within a picture, and *Extract* means extract hidden text from a picture. You can return to this slide at any time by clicking on *Cancel*, and you can reset everything by selecting *Start Over* from the *File* menu.

This is the initial state of the walkthrough when the program is originally loaded and after *File* ⇉ *Start Over* has been chosen. Once the walkthrough knows which process is to be carried out, it can offer hints.

State 2:	Next (3)	Back (1)	Cancel (1)

Choose an image to use as a key by selecting it on the image bar below, and clicking the load button on the Key Image pane. Other people will be able to extract the text from the picture if they use the same Key Image. If you want to use an image from another directory, you can use *Set Image Directory* from the *Tools* menu.

The simplest technique for loading a key is described.

State 3:	Next (4)	Back (2)	Cancel (1)

Please pick out the Target Image that you would like to hide the text in by selecting it on the image bar and clicking the load button on the Target/Source Image pane. Again, for images from another directory, select *Set Image Directory* from the *Tools* menu.

The simplest technique for loading a target image is described.

State 4: Embed (5) Back (3) Cancel (1)

Enter text into the text box on the Text tab of the main panel. You can also paste text from the clipboard with the *Paste* command in the *Edit* menu, or you can choose *Open Text* from the *File* menu. When you are done, click the *Next* button to hide the text in the source image.

The user is instructed to input text using combinations of three different methods: typing, pasting, and importing.

If the text cannot be embedded for some reason, an appropriate dialog will appear informing the user what he or she can do to alleviate the problem.

State 5: Next (0) Back (4) Cancel (1)

Click on the *Image* tab on the main panel to see the resulting image. If you are satisfied with its appearance, choose *Save Image* or *Save Image As…* from the *File* menu.

After viewing the image, the user saves it if satisfied. Otherwise, he or she can click *Cancel*. After *Save* or *Save As…* is selected, the *Back* and *Cancel* buttons should preferably be grayed out.

State 10: Next (11) Back (1) Cancel (1)

Choose an image to use as a key by selecting it on the image bar below, and clicking the load button on the Key Image pane. If you want to use an image from another directory, you can use *Set Image Directory* from the *Tools* menu.

The simplest technique for loading a key is described.

State 11: Extract Text (12) Back (11) Cancel (1)

Please pick out the image from which the hidden text should be extracted. Select it on the image bar and click the load button on the Target/Source Image pane. Again, for images from another directory, select *Set Image Directory* from the *Tools* menu.

The simplest technique for loading a source image is described.

If the text cannot be extracted for some reason, an appropriate dialog will appear informing the user what he or she can do to alleviate the problem.

State 12: Next (0) Back (12) Cancel (1)

The text can be exported to another application via the clipboard by selecting *Copy* from the *Edit* menu. Alternatively, the text can also be exported to a text file by selecting *Save Text* from the *File* menu. Otherwise the text can be read as it stands by the eyes it was intended for!

The user is instructed as to how the extracted text can be exported.

State 0: Next (gray) Back (gray) Cancel (gray)

To Embed or Extract again, select *Start Over* from the *File* menu.

After a process has been completed, the walkthrough is at an end. All of the walkthrough buttons are grayed out, until it is restarted by choosing *Start Over*.

3 FUNCTIONALITY DETAILS

The fine details of the functionality that the end user should expect are given in this section. Main points are in **bold** with bullets, and sub-points are in normal text.

- **Platform Independence**
  The program will use Java and will be platform independent.

- **Start, Exit and *Minimize* the application**

Start: The user invokes the program from the operating system with the aid of the Java virtual machine (VM). The program is initialized with no open files (i.e., no keys, images, etc.)

Exit: If the user selects Exit from the File menu, the program will begin exiting. If any extracted text has not been saved, or if an embedded image has not been saved, a dialog will pop up asking the user, if he or she wants to save the relevant data.

Min: If the user minimizes the window, it will be minimized by the operating system. Should the window be reactivated again, the program will return to the same state as before it was minimized.

- **Start Over**
  If the user selects Start Over from the File menu, then the program should return to its initial state (i.e., the same state as when the program was initially started). If any extracted text has not been saved, or if an embedded image has not been saved, a dialog will pop up warning the user or asking if he or she wants to save the relevant data.

- **Import/Edit Text**

File: Using the option Open Text from the File menu, text can be loaded from a file into the large text box. A dialog will pop up asking the user to indicate the input file.

Kybd: The keyboard can also be used to enter text directly into the text box.

Paste: The Paste command from the Edit menu can be used to import text from the clipboard.

Edit: The text can be edited using the standard Copy, Cut, and Paste commands from the Edit menu.

Scroll: If the text is longer than the box, then the box should be scrollable so that the user can see and edit the entirety of the text.

- **Export Text**
  The same as for Import Text.

- **Open Image**
  As long as there are no dialogs open, a new target/source image can be loaded at any time. This can be accomplished by selecting Open Image from the File menu. A dialog will be used to help select the file. Otherwise, the image can be selected from the image bar and loaded onto the Target/Source Image box in the main window. A scaled version of the current loaded image will always be present in the Target/Source Image box, so that the user can be reminded of what he or she is working with. Multiple file formats will be supported for opening.

- **Save Image and Save Image As…**

Save: If the user selects Save from the File menu, then the image will be saved to its default location. If this location is unknown, then the Save As… dialog will appear asking the user where to save the image. The Save As… dialog will also be needed if the input image is in a format that cannot be saved. At least the GIF format will be supported for saving. The user should additionally be instructed if overwriting files.

Sv. As: Selecting Save As… from the File menu will cause a dialog to pop up asking where the image should be saved. The user should be instructed if overwriting files.

- **Manage Keys**

St key: If no key is loaded using the following methods, a default key, which is built into the application, will be available for extracting and embedding text. The text representation of the key will be in ASCII using hexadecimal (i.e., with the digits 0-F).

Image: A key can be generated by hashing a key image. This can be accomplished by selecting Open Image Key from the Tools menu. A dialog will be used to help select the file to be used to generate the key. Alternatively, the user can select an image on the image bar and load it into Key Image box on the right side of the window. A scaled version of the current key image will always be present in the Key Image box, so that the user can be reminded of what he or she is working with.

Kybd: If the user selects the Text tab on the Key Image panel, he or she may enter a text version of the key. If the current key is the default key, the text box will be empty so that the default key will not be shown. A key incompatible with the algorithms involved will not be accepted.

TxtLd: If Open Text Key is chosen from the File menu, then the user will be able to choose a text file from which a key will be loaded. If the key is invalid, the user will be informed and the key will not be loaded.

TxtSv: Finally, if Save Text Key is chosen from the File menu, the user will be able to choose a location where the current key should be saved.

- **Walkthrough**

There is walkthrough, which assists and teaches the user how to use *Paranoia.* While the program can be used independently of the walkthrough, it is easier to carry out the general embed and extract processes by following the walkthrough.

- **Help Menu**

HTML files will provide instructions to the user.

- **Embed Text in an Image**

Menu: If selected, the text from the text box will be encrypted and embedded into the target image using the current key. The target image will be changed slightly during the process. If the Current image in the Image box has not been saved a confirmation dialog will be shown.

Wlkth: In a certain stage of the walkthrough, clicking next will invoke the same code as selecting Embed Text from the Tools menu.

- **Extract Text from an Image**

Menu: If Extract Text is selected, *Paranoia* will attempt to extract text out of the source image using the current key and write it into the main text box replacing the contents. A confirmation dialog will appear if unsaved text is present in the main text box. If the operation is unsuccessful (i.e., the key was incorrect), then the user will be informed and the text in the main text box will not be changed.

Wlkth: In a certain stage of the walkthrough, clicking the Next button will invoke the same code as selecting Extract Text from the Tools menu.

- **Viewing the Text and Viewing the Target/Source Image**

Text: The current text in the main text box can be viewed by selecting the appropriate tab in the main window.

Image: An unscaled version of the output image can be viewed by selecting the appropriate tab in the main window. If it is larger than the window, then there will be an obvious method for viewing other parts of the image.

Image: A scaled version of the current target/source image will also always be shown in the Target/Source Image box in the main window.

4 GUI SPECIFICIATION

The user interface for this program will be provided with the following graphical features:

4.1 THE MENU BAR

File

The File menu provides the user with the basic functions that are expected to be found in a File menu, plus a few more. Namely, the following options can be accessed through this menu:

- *Start Over:* Frees the current workspace so user can restart without any loaded documents or work in progress.
- *Open Text:* This menu item simply opens a file dialog, which allows the user to load a text file into the main text box.
- *Save Text:* Allows user to save the text in the main text box; selectable if text is present in main text box.
- *Save Text As…:* Allows user to save the text in the main text pane as a different file than what is was opened from.
- *Open Image:* Opens an image into the Target/Source Image Pane, which can be used to extract text from or to embed text into.
- *Save Image:* Allows user to save generated Image with embedded text, located in main image box.
- *Save Image As…:* Allows user to save generated Image with a different file name than it already has.

Edit

The Edit menu allows the user to manipulate text in the text area that has focus. The user can select a block of text by left clicking the beginning of the block of text, and dragging the mouse to the end of the block. There are three options in the Edit menu:

- *Cut:* Removes the selected text from the document and puts it into the clipboard, replacing previous clipboard contents.
- *Copy:* Puts the selected text into the clipboard, replacing previous clipboard contents.
- *Paste:* Inserts the text on the clipboard into the document, beginning from the current position of the text cursor.

Tools

The Tools menu allows the user to manage the program. The options in the Tools menu will be selectable depending on whether necessary boxes of the program have been filled. The text of a non-selectable menu option will appear faded in comparison to the text of a selectable menu option to indicate that the user cannot choose the faded option.

- *Embed Text:* Embeds the text in the Main Panel into the selected image in the Target Pane. Selectable when:
  - A valid target image has been chosen, and
  - Text has been entered in the Main Panel

  Not selectable otherwise.

- *Extract Text:* Extracts text from an image and displays it in the Main Panel. Selectable when:
    - An image has been loaded into the Target/Source

  Not selectable otherwise.
- *Open Image Key:* Opens a file browser to select an image to use as a key. When the user selects an image, it will appear in the Key Pane.
- *Open Text Key:* Opens a file browser to select a text file to use as a key. When the user selects a text file, its contents will appear in the Key Pane.
- *Save Text Key:* Saves the text key in the Key Pane by opening a file browser to select a location to save the text file. It is usable when text has been entered in the Key Pane.

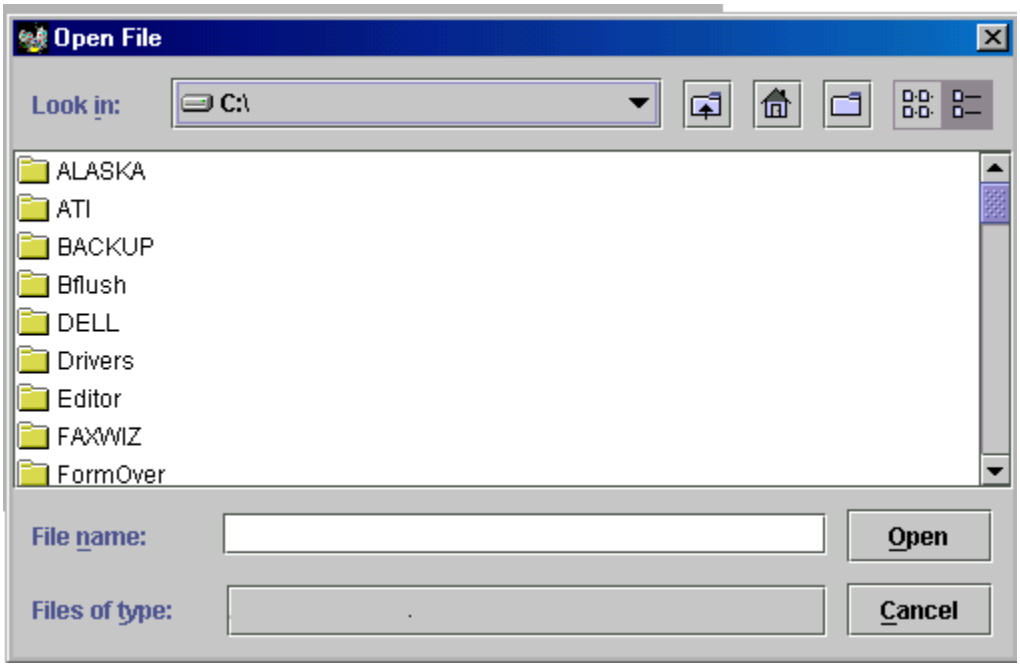
Submenu: Tools ⇶ **Options**

Encryption ON/OFF:

This menu option allows the user to choose whether or not the text should be encrypted before being hidden within the image or not. By default, encryption is on.

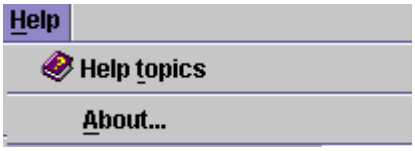
Set Image Directory:

This menu prompts user for setting the Image Directory, the directory where the images displayed on the image thumbnail bar are stored.
When pressing the Set Image Directory menu item, the user is shown a standard open file dialog where he or she can choose the directory for the images displayed on Image Bar.



Help

The Help menu contains two items: Help Topics and About….



The help topics item opens the system's default browser and displays various help topics related to the program's usage in HTML format.

The about item opens a window describing the team of the application builders and lists related credits.

4.2 GUI COMPONENTS

Main Panel

The main panel of the application contains a Text tab and an Image tab as described below. Both the text and image cannot be viewed at the same time in the main panel; however, a scaled version of the image is always shown in the Source/Target pane, so both will be visible if the text tab for the main panel is selected.

- *Text Tab:* This is the main text area into which the user types the text to be hidden within an image. In the case of extracting text from an image, the text would be loaded into this same tab.
- *Image Tab:* This tab is also located within the main panel. Its sole purpose is to display an image that has been produced as a result of embedding text within another image.

Key Pane

The program supports two types of keys: text and image. The Key Pane will have two corresponding tabs that will display the selected key. The selected tab will be the key the program uses. For example, even if a key is selected in the image tab and a key has not been selected in the text tab, the program will not use the image key if the text tab is selected.

- *Image tab:* To use an image as a key:
  - The Key Pane has a button that allows the user to use the selected image in the Image Bar as an image key. This button will be active when the image tab is active and an image in the Image Bar has been selected.
  - The user can also use the Open Image Key option in the Tools menu bar to load an image as a key.
- *Text tab:* The user may type directly into the text tab. He can also use Open Text Key from the Tools menu to load text as a key. The text continues to be editable once a text file has been loaded.

Source/Target Pane

The **Source/Target** Pane shows the image that will contain hidden text, or an image, which we would like to extract text from.
An image can be loaded into the Source/Target Pane either from the Image Bar via the Load Image button that will load highlighted image or through File menu ⇉ Open Image File.

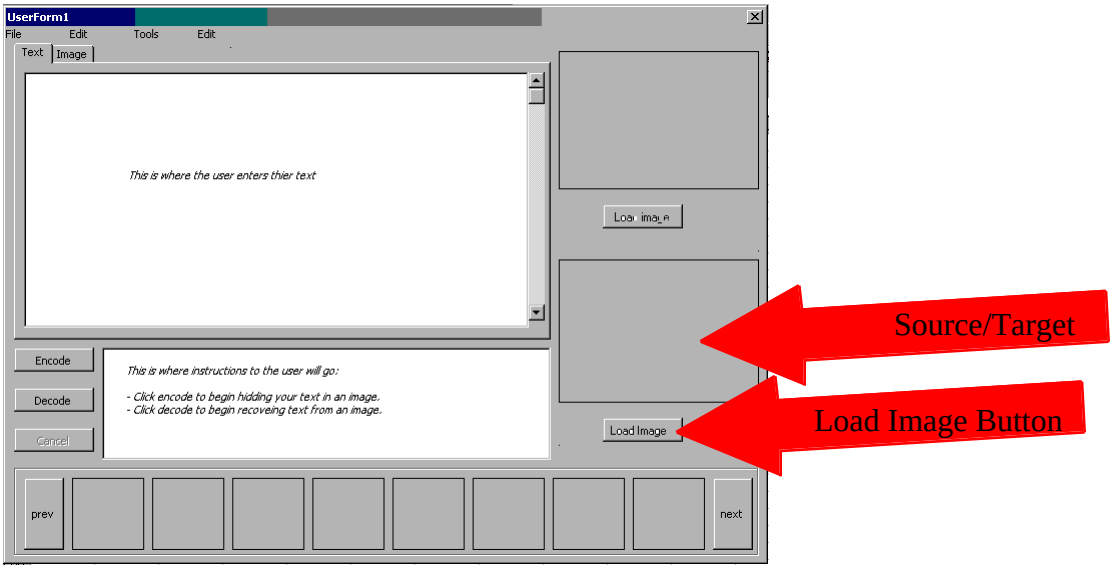
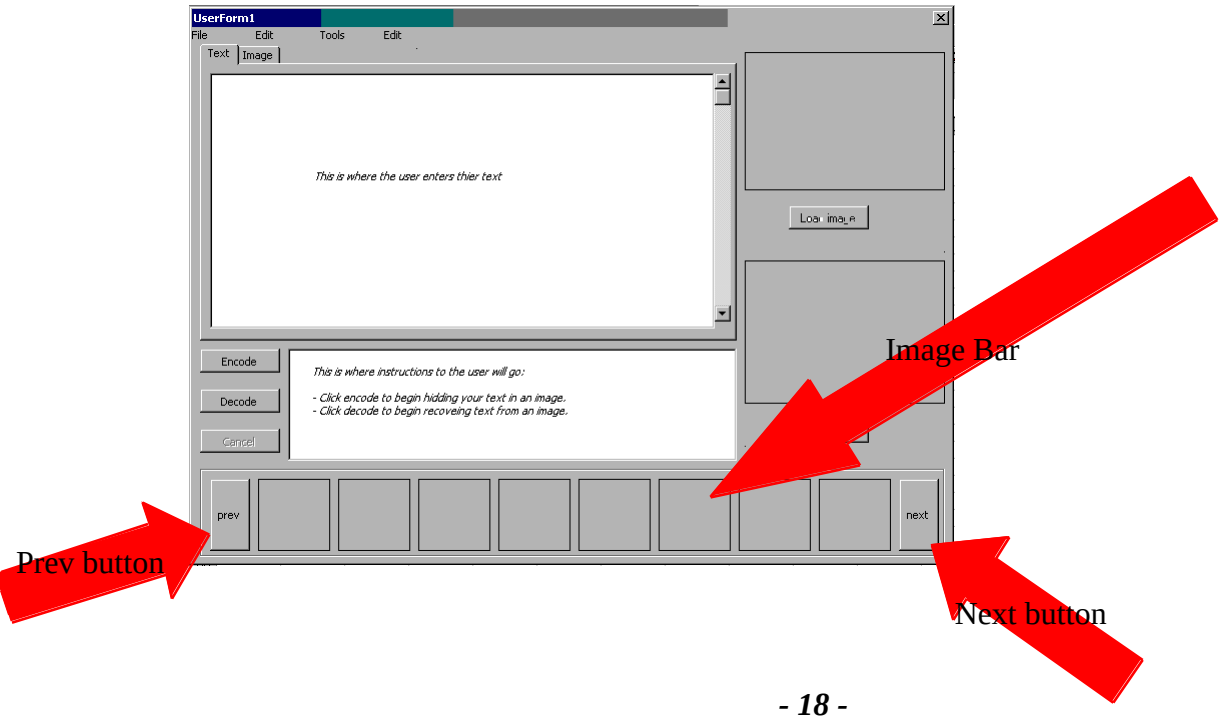


Image Bar

The Image Bar will show thumbnails of images within the selected image directory (chosen through Tools ⇉ Options ⇉ Set Image Directory). The user will be able to navigate through the bar using Previous and Next buttons.



Walkthrough Navigation

As seen in the above images, the application also has an easy to use walkthrough system for novice users. The three buttons in the mid-section of the program, located to the left of the directions box, help a user through an *embedding* or *extracting* session, guiding the user with advice displayed in the text area to the immediate right.

Appendix A: GLOSSARY

Key Image

An image used by *Paranoia* to generate keys used to implement its security features. This image is not altered in the process.

Source Image

An image, which is believed to contain hidden text. It is indistinguishable from a regular image, until extraction is preformed.

Target Image

An image from which the user would like to produce a copy, or output image, which looks as similar as possible but still contains hidden text.

Output Image

An image which has had text embedded into it, and can be exported or save to the user's hard drive.

Keys

The input to *Paranoia*'s security algorithms. They are in one of three forms; an image, a text string, or an internal default key used if the user does not supply one.

Pane / Panel

An area of the screen pertaining to a particular control or button.

Extract

The process of finding and recovering text hidden in an image file.

Embed

The process of hiding text in an image file.