ISE Test Plan

March, 2004

Team ISE

Image Selective Encryption

CSCI 4308-4318, Software Engineering Project
Department of Computer Science
University of Colorado at Boulder

Shinya Daigaku
Geoffrey Griffith
Joe Jarchow
Joseph Kadhim
Andrew Pouzeshi

Sponsored by:
Tom Lookabaugh
Assistant Professor of Computer Science

Project Proposal

Traffic constantly flows between computers connected to the Internet. Large volumes of information may take a long time traveling from destination to destination. Such a reduction in speed makes it desirable to compress the file as much as possible in order to send the smallest amount of data required. Thus, compression of data has allowed for the high-speed data transfers that have made Internet communication and business more feasible.

In addition to sending the smallest amount of information possible, users also attempt to maintain a certain level of security upon their information. Due to the fact that common encryption methods generally manipulate an entire file, most encryption algorithms tend to make the transfer of information more costly in terms of time and bandwidth. Thus, users pay a price for security relative to their desired level of security. One possible solution would be a system of encryption that works cooperatively with the standard compression schemes. *Selective Encryption* of only a small percentage of the file's bits will facilitate this solution. Because most encryption schemes will make the file larger, selective encryption seeks only to encrypt portions of the file that will make it unusable. In other words, if a user does not have the proper decryption device, the file should not be usable. Selective encryption will minimize the necessary increase in file size due to encryption while maintaining a maximum level of uselessness, or damage, to the product.

Team ISE (Image Selective Encryption) will deliver a package for selectively encrypting JPEG (Joint Photographic Experts Group) still image files. The package will provide the tools necessary to encrypt the critical information of a JPEG file in cooperation with existing standard compression tools. This package will handle JPEG files in such a way that only a small percentage of the total file will be encrypted. Selective Encryption security will not extend to the level of complete encryption, but rather to a level that would deter all but brute force attacks, allowing users to securely protect private JPEG images.

A JPEG image could be encrypted with any of the sufficiently secure encryption algorithms available to the open source community, but this can result in an increase in file size or can require a large amount of processing time. However, by selecting small but vital portions of a file and encrypting only those few bytes can render an image unusable. The initial statistical analysis done by the team will consist of specifically breaking down the standard JPEG compression scheme into its usable parts and evaluate which of the parts, if encrypted, will cause a potential user to pay for rights to the image or force subscription to the provider service.

An additional aspect of the encryption analysis will be the determination of the specific targets in the file for encryption. For example in an MPEG file there are headers that contain a small portion of the overall number of bits but which are extremely vital to the reproduction of the movie by the user. So, if certain headers were to be encrypted the percentage of the file being manipulated would be less than ten percent of the total number of bits in the file. Although only a small portion will be encrypted, the resulting damage experienced by an unauthorized user would be sufficient to cause the user to pay for the decryption package. However, there are other targets that, while they can be encrypted and will do sufficient damage, can be guessed by an

attacker. The attacker could, with some degree of effort, render the file useful without use of the decryption software. For example, if the frame rate of an MPEG file was encrypted, an attacker could try all three of most common frame rates and one of these is certain to produce the correct rate for the particular video. In the case of JPEG Selective Encryption, Team ISE will have to balance the targets for encryption against ease of simple attacks.

A permanent web site will be constructed by the team to make the software package available to anyone interested in the Team's project. As it is vital to the world of cryptography to let the community view the approach, the first form of the working prototype will be made available on the web site. From this, feedback can be received not only from the team itself, but also from the cryptography community at large.

So, following the guidelines of the ongoing MPEG research (also being guided by the sponsor), the team will study the JPEG process and earlier attempts at encryption. With the sponsor's assistance, Team ISE will devise a workable approach to handling individual JPEG images following the concept of selective encryption.

1. INTRODUCTION

Team ISE is sponsored by Assistant Professor of Computer Science, Tom Lookabaugh, at the University of Colorado: http://itd.colorado.edu/lookabaugh/. Tom Lookabaugh is currently involved in selective encryption research on standard MPEG (Moving Picture Experts Group) files and is interested in researching the application of Selective Encryption for other multimedia formats.

The goal of selective encryption is to minimize the amount of encryption applied to a file while maximizing the damage done to the image being viewed by a user not in possession of the authorized decryption package. Complete encryption is not a requirement of the process, nor is rendering the file useless to the level of complete military secrecy. It is acceptable for an attacker to be able to view portions of the file; however, the file should be distorted enough that an attacker would not wish to use the encrypted file, but would rather purchase or subscribe to the decryption method for access to the original files.

Multimedia files prove to be good subjects for selective encryption, as these files tend to be very large and employ compression algorithms that concentrate critical information in small portions of their bit stream. If the critical data in certain multimedia standards is encrypted properly, the remaining information becomes useless to those without the appropriate decryptor. There are many types of compression algorithms that fit this description, such as MPEG 1, 2 and 4 video, G.723 and G.729 video, AAC audio, MP3 audio, JPEG and JPEG2000 image formats. Applying a Selective Encryption security solution to selected multimedia formats will greatly increase the protection level of important information.

The focus of the ISE project is to research and develop an algorithm for selectively encrypting the JPEG *baseline* compression image standard. The product of the research and development will be a package that will encrypt a file so that the amount of the file being encrypted is relatively small (on the order of 1-2% of the total file). The product will be delivered in a package that will include an encryptor and a decryptor for JPEG files and a testing suite. A web site will be constructed to facilitate the delivery of the product and documentation about the process. The encryptor and decryptor will encrypt and decrypt selected targets contained within JPEG files. The ISE project will employ the AES (Advanced Encryption Standard) for our Selective Encryption algorithm. This package will be made available in a purely open source form on our final web site.

In addition to the package containing the decryptor and encryptor, Team ISE will also provide a test suite available to prospective users. The test suite will be used to aid in the research, development and testing of the team's final product. The test suite will provide the functions necessary to complete this project. First, it will allow the user to preview a standard JPEG image. Second, the test suite will break down the various portions of a JPEG image and provide the ability to manipulate the data in all of the portions. Third, after altering the data in any particular file, the test suite will provide the capability to preview the encryption attempt without the benefit of compatible decryption. Forth, the suite will have the ability to decrypt an encrypted file. The decryption options will allow the user try to defeat the encryption methods.

Any selective encryption scheme could be developed using a package that implemented these features, however, the delivered test suite will only employ the AES encryption scheme chosen by the team. The test suite will be available to download from the team web site.

The final web site will be deployed on a web server provided by the Sponsor. The machine facilitating the web server will use the Linux Red Hat 9.0 operating system platform. The team will acquire a fixed IP address from the proper University of Colorado authorities and will develop a simple web site capable of delivering information to viewers about the benefits and application of Selective Encryption technology. The site will provide users the option to download and use the final software package. The site will also provide links to important information and will remain in place as long as the sponsor deems necessary.

The final software package will accomplish the complex task of selectively encrypting a JPEG baseline standard image while providing a simple user interface. Team ISE has identified three specific types of users: high-end art users, typical Internet image users, and small, low-end image users. The research and software will be tailored to these users' needs. Figure 1.1 is a flow chart showing the general logic design of the team's final product.

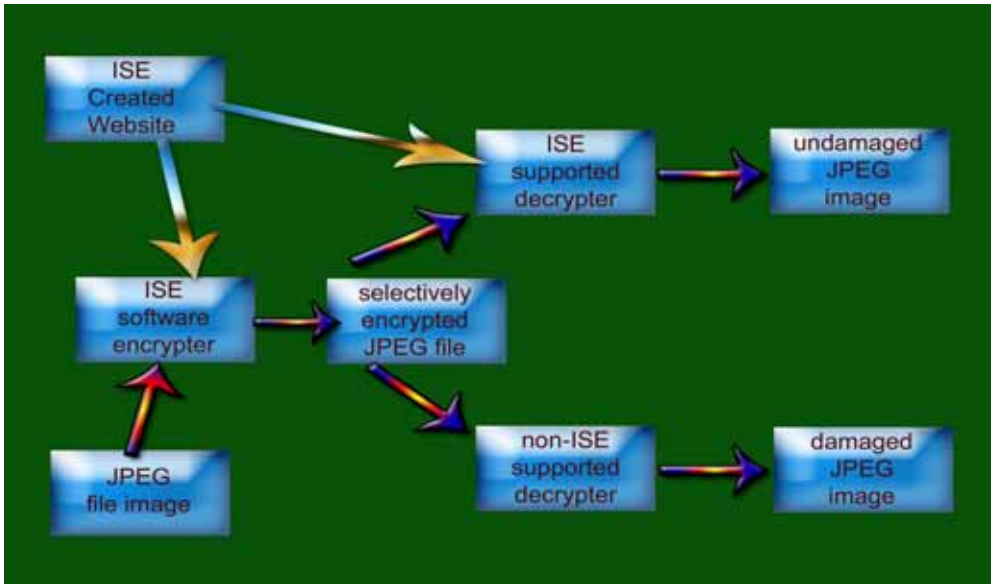

Figure 1.1: Conceptual Overview of ISE Software

This document describes the test plan for the various components of the ISE project, and is used to verify that the project meets the requirements set forth in the ISE *Requirements Document*. It describes the environments, both hardware and software, necessary to test the production code, Manipulator, and web site. It then proceeds to give a detailed description of the tests themselves.

2. TEST ENVIRONMENT

This section of the text plan document outlines that Environment used to test the ISE Production Code, the ISE Manipulator, and the ISE web site. The ISE Production Code tests should be conducted in the following environment:

Software:

- Any Version of Redhat Linux 9.0 and higher.
- Windows 9x/ME/NT/200x/XP and higher.
- Mac OS X or higher.

Hardware:

- Generic Color Monitor.
- Mouse as part of the User Interface.
- Keyboard as part of the User Interface.
- Support for a 32-bit processor assembly instructions for AES optimizations.

The ISE Manipulator tests should be conducted in the following environment:

Software:

- Windows 9x/ME/NT/200x/XP and higher.
- Microsoft .NET Framework Version 1.1 or Higher.

Hardware:

- Generic Color Monitor.
- Mouse as part of the User Interface.
- Keyboard as part of the User Interface.

The ISE Web Site tests should be conducted in the following environment:

Software:

- Microsoft Internet Explorer 6.0 or higher.
- Netscape Navigator 6.0 or higher.
- Mozilla 1.5.1 or higher.
- Safari 1.0 or higher.
- Support for HTML version 4.01 transitional.

Hardware:

- Generic Color Monitor.
- Mouse as part of the User Interface.
- Keyboard as part of the User Interface.

Unless explicitly invoking any instance of the ISE products as part of a test procedure, these tests assume that an instance of the product is running.

3. TESTS

The tests are organized into three separate sections which deal with the different components of the ISE project. The sections are:

1. **ISE Production Code**
2. **ISE Manipulator**
3. **ISE Web Site**

Each test in the Test Plan has seven components:

Purpose	The reason for the test.
Procedure	The steps to follow to conduct the test.
Expected Result	The results necessary to pass the test.
Comments	Any comments the tester might have.
Date	Date the test was conducted.
Tester	Name of the person conducting the test.
Outcome	Outcome of the test (Pass or Fail).

3.1. Production Code Test

This section of the test plan is to outline out all of the testing requirements and desired results for the ISE Production Code. To test all of the functionality provided by this class, we've have designed a set of tests to cover all of the class methods. These tests were conducted as outlined in the following sections.

3.1.1. JPEG_ISE Constructor with Key Only

Purpose:	The purpose of this test is to determine if a jpeg_ise object can be created with only an encryption/ decryption key.
Procedure:	1. Create a pointer to a character array in a C++ program containing the desired key information. 2. Call the jpeg_ise(key) constructor with this key as the only parameter.
Expected Result:	A new object of type jpeg_ise will be created. The key will be set using the information passed in the parameter. A default value of NULL will be set for both the input and output file names.
Comments:	In order to use this object for encryption or decryption, the user must call the set_input_file_name() and set_output_file_name() functions to set the desired jpeg and ise files.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.2. JPEG_ISE Constructor with All Parameters

Purpose: The purpose of this test is to determine if a jpeg_ise object can be created with an encryption/ decryption key as well as the input and/or output file name.

Procedure: 1. Create three pointers to character arrays in a C++ program, the first containing the desired key information, the second containing the input file name and the output file name.
2. Call the jpeg_ise() constructor with all three pointers as it's arguments.

Expected Result: A new object of type jpeg_ise will be created. The key will be set using the information passed in the first parameter. The second and third parameters will be used to set the input and output file names.

Comments: For the input and output file names, one parameter should be a jpeg file name and the other should be an ise file name, in either order. The user can verify that the input and output files were set correctly using the get_input_file_name() and get_output_file_name() functions.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: **Pass**

3.1.3. Set_Key Function with Valid Key

Purpose: The purpose of this test is to determine if a key can be created with a valid character string.

Procedure: 1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing one or more characters indicating the desired key information.
3. Call the set_key() function with this key as the only parameter.

Expected Result: The encryption/decryption key will be created for the object using the information in the character array. The function should return 0 to indicate that the key was successfully created for the object.

Comments: The key information in the calling program should not be damaged or modified in any way by this function.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.4. Set_Key Function with Invalid Key

Purpose: The purpose of this test is to determine if the set_key() function exits gracefully given NULL for the key information.

Procedure: 1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing NULL, which is invalid for jpeg_ise key information.
3. Call the set_key() function with this key as the only parameter.

Expected Result: It should return 1 indicating an invalid key.

Comments: If the object did not contain a valid key previous to this function call, the function will need to be called again with a valid key for the object to be used for encryption or decryption.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.5. Set_Input_File_Name Function with Valid Input File

Purpose: The purpose of this test is to determine if an input file name can be created with a valid character string.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing the desired input file name with a .jpeg, .jpg, or .ise extension.
3. Call the set_input_file_name() function with this pointer as the only parameter

Expected Result: The input file name will be created for the object using the information in the character array. The function should return 0 to indicate that the input file name was successfully created for the object.

Comments: The input file name information in the calling program should not be damaged or modified in any way by this function. The input file must exist and be of either ise or jpeg type.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.6. Set_Input_File_Name Function with NULL

Purpose: The purpose of this test is to determine if the set_input_file_name() function exits gracefully given NULL for the file name.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing NULL for the input file name.
3. Call the set_input_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's input file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid input file name previous to this function call, the function will need to be called again with a valid file name for the object to be used for encryption or decryption.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.7. Set_Input_File_Name Function with Non-Valid File

Purpose: The purpose of this test is to determine if the set_input_file_name() function exits gracefully given a non-valid file for the file name, i.e. the file is of neither jpeg nor ise type.

Procedure

1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing a non-valid file for the input file name. Examples of non-valid files are bitmaps, text files, or any other non-jpeg or non-ise file types.
3. Call the set_input_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's input file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid input file name previous to this function call, the function will need to be called again with a valid file name for the object to be used for encryption or decryption.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.8. Set_Output_File_Name Function with Valid Output File

Purpose: The purpose of this test is to determine if an output file name can be created with a valid character string.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing the desired output file name with a .jpeg, .jpg, or .ise extension.
3. Call the set_output_file_name() function with this pointer as the only parameter.

Expected Result: The output file name will be created for the object using the information in the character array. The function should return 0 to indicate that the output file name was successfully created for the object.

Comments: The output file name information in the calling program should not be damaged or modified in any way by this function. The output file must exist and be of either ise or jpeg type.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.9. Set_Output_File_Name Function with NULL

Purpose: The purpose of this test is to determine if the set_output_file_name() function exits gracefully given NULL for the file name.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing NULL for the output file name.
3. Call the set_output_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's output file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid output file name previous to this function call, a default name will be created during encryption or decryption based on the input file name.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.10. Set_Output_File_Name Function with Non-Valid File

Purpose: The purpose of this test is to determine if the set_output_file_name() function exits gracefully given a non-valid file for the file name, i.e. the file is of neither jpeg nor ise type.

Procedure:
1. Create a jpeg_ise object.
2. Create a pointer to a character array in a C++ program containing a non-valid file for the output file name. Examples of non-valid files are bitmaps, text files, or any other non-jpeg or non-ise file types.
3. Call the set_output_file_name() function with this pointer as the only parameter.

Expected Result: The function should exit without setting the jpeg_ise object's output file name. It should return 1 indicating an invalid file name.

Comments: If the object did not contain a valid output file name previous to this function call, a default name will be created during encryption or decryption based on the input file name.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.11. Get_Input_File_Name Function When input_file_name != NULL

Purpose: The purpose of this test is to determine if the get_input_file_name() function returns the proper string indicating the name of the input file.

Procedure: 1. Create a jpeg_ise object with a valid input file.
2. Call the get_input_file_name() function with no parameters.

Expected Result: The function should return a pointer to a character string containing the same name as the input file used when creating the object.

Comments: If the input file is properly set for the jpeg_ise object, then a valid pointer to the char array will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.12. Get_Input_File_Name Function When input_file_name == NULL

Purpose: The purpose of this test is to determine if the get_input_file_name() function returns NULL when the input_file_name is equal to NULL.

Procedure: 1. Create a jpeg_ise with key only.
2. Call the get_input_file_name() function with no parameters.

Expected Result: The function should return NULL.

Comments: If the input file is not explicitly set by the user for the jpeg_ise object, then the default NULL will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.13. Get_Output_File_Name Function When input_file_name != NULL

Purpose: The purpose of this test is to determine if the get_output_file_name() function returns the proper string indicating the name of the output file.

Procedure:
1. Create a jpeg_ise object with a valid output file.
2. Call the get_output_file_name() function with no parameters.

Expected Result: The function should return a pointer to a character string containing the same name as the input file used when creating the object.

Comments: If the output file is properly set for the jpeg_ise object, then a valid pointer to the char array will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.14. Get_Output_File_Name Function When input_file_name == NULL

Purpose: The purpose of this test is to determine if the get_output_file_name() function returns NULL when the input_file_name is equal to NULL.

Procedure:
1. Create a jpeg_ise object with key only.
2. Call the get_output_file_name() function with no parameters.

Expected Result: The function should return NULL.

Comments: If the output file is not explicitly set by the user for the jpeg_ise object, then the default NULL will be returned.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.15. Encrypt_File Function Normal Use

Purpose: The purpose of this test is to determine if the encrypt_file() function selectively encrypts a jpeg image.

Procedure:
1. Create a jpeg_ise object with a valid key, input jpeg file, and output ise file.
2. Call the encrypt_file() function with no parameters.

Expected Result: The function should return 0 to indicate success. The original jpeg image should be undamaged and the ISE file should contain the encrypted jpeg.

Comments: The function should return 0 to indicate success. The original jpeg image should be undamaged and the ise file should contain the encrypted jpeg.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.16. Encrypt_File Function with Invalid Input File

Purpose: The purpose of this test is to determine if the encrypt_file() function exits gracefully given an input file that does not exist.

Procedure:
1. Create a jpeg_ise object with a valid key and output ise file name and a jpeg file name that does not exist.
2. Call the encrypt_file() function with no parameters.

Expected Result: The function should return 1 to indicate that the input jpeg file could not be opened. The function should then exit without encrypting any data.

Comments: The output ise file should be empty due to the fact that no encryption was performed.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.17. Encrypt_File Function with Output ISE File Name Not Set

Purpose: The purpose of this test is to determine if the encrypt_file() function calls the make_ise_file_name() function to make a default output ise file.

Procedure: 1. Create a jpeg_ise object with a valid key and input jpeg file. Leave the output file name to be the default NULL.
2. Call the encrypt_file() function with no parameters.

Expected Result: The function should call make_ise_file_name() to create an ise file name based on the input jpeg file name. Encryption should proceed and return 0 indicating a success.

Comments: The output ise file should be created and named based on the input jpeg file. This file will contain the encrypted jpeg file information. If the ise file could not be created for any reason, this function will return 2.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.18. Decrypt_File Function Normal Use

Purpose: The purpose of this test is to determine if the decrypt_file() function selectively decrypts an ise image.

Procedure: 1. Create a jpeg_ise object with a valid key, input ise file, and output jpeg file.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should return 0 to indicate success. The original ise image should be undamaged and the new jpeg file should contain the exact same information as the original jpeg.

Comments: To test if the image decrypted properly, the user can try to look at the image. Also, to make sure that there is no difference between the original and decrypted jpeg images, the user could run the Unix “diff” command on the two files.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.19. Decrypt_File Function with Non-JPEG-ISE Input File

Purpose: The purpose of this test is to determine if the decrypt_file() function exits gracefully given an input ise file that is not an encrypted jpeg image, i.e. the ise file contains a decrypted mp3 or zip file.

Procedure:
1. Create a jpeg_ise object with a valid key and output jpeg file name and an ise file name that contains an encrypted mp3 or zip file.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should return 5 to indicate that the input file is not jpeg-ise. The function should then exit without decrypting any data.

Comments: Due to the fact that the only ise files that exist are all from jpegs, the tester will have to change the first byte in the ise to mimic a different ise file type.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.20. Decrypt_File Function with Invalid Input File

Purpose: The purpose of this test is to determine if the decrypt_file() function exits gracefully given an input ise file that does not exist.

Procedure:
1. Create a jpeg_ise object with a valid key and output jpeg file name and an ise file name that does not exist.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should return 2 to indicate that the input ise file could not be opened. The function should then exit without decrypting any data.

Comments: The output jpeg file should be empty due to the fact that no decryption was performed.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.21. Decrypt_File Function with Output File Name Not Set

Purpose: The purpose of this test is to determine if the decrypt_file() function calls the make_output_file_name() function to make a default output file.

Procedure:
1. Create a jpeg_ise object with a valid key and input ise file. Leave the output file name to be the default NULL.
2. Call the decrypt_file() function with no parameters.

Expected Result: The function should call make_output_file_name() to create an output file name based on the input ise file name. Decryption should proceed and return 0 indicating a success.

Comments: The output jpeg file should be created and named based on the input jpeg file. This file will contain the decrypted jpeg file information. If the ise file could not be created for any reason, this function will return 2.

Date: March 6, 2004

Tester: Joe Jarchow / Joseph Kadhim / Shinya Daigaku

Outcome: Pass

3.1.22. Decrypt_File Function with Incorrect Key

Purpose The purpose of this test is to make sure that the decrypt_file() function does not produce a properly decrypted jpeg image when given an incorrect key.

Procedure:	1. Encrypt a jpeg image and create an ise file with a valid key 2. Call the set_key() function with a new valid key. 3. Call decrypt_file with the ise file and the new key.
Expected Result:	The function should return 0 to indicate that the file was decrypted. The new jpeg image produced should not be a valid jpeg image.
Comments:	To test that the image did not decrypted properly, the user can try to look at the new image. Also, the user could run the Unix "diff" command on the original image and the new decrypted image to see that there are differences.
Date:	March 6, 2004
Tester:	Joe Jarchow / Joseph Kadhim / Shinya Daigaku
Outcome:	Pass

3.2. Manipulator Test

This section of the test plan is to list out all of the testing requirements and desired results for the ISE JPEG Manipulator. To test this massive amount of functionality, this testing breaks down into three main pieces:

1. Menu Options
2. Button Controls
3. General Tests

The "Menu Options" section will test all of the different menu options available in the Manipulator, like the "Save Project" or "Open Picture" options that are available. The "Button Controls" section will test all of the different button control found within the Manipulator, like "Save Project" or "Update Picture" buttons available on the Project sub-tab on the Console tab. Finally, the "General Tests" section of this document will test all the rest of the miscellaneous functionality, like if the SEP project file is set up correctly or to test if the TextBox controls are working correctly.

3.2.1. Menu Options

This section of the test plan is to list out the menu functions that need to be tested. Included in this section is each of the tests, a short description of the test and the expected results.

3.2.1.1. File Menu Tests

This section of the test plan is to test all of the File Menu options. Each of the File Menu options has a test under this section.

3.2.1.1.1. New Project Menu Option Test

Purpose: To test the "New Project" menu option to make sure that a new project is created when this option is selected.

Procedure: 1. Prior to choosing the "New Project" option, open a new picture in the Manipulator.
2. Then click the "New Project" menu option under the File menu.

Expected Result: All of the old information in the Manipulator should be cleared out for a new project to be created and they should be prompted for a new project file name and path.

Comments: This is not required to make a new project, for instance, you could just load in a picture and then click the "Save Project" option and the current information loaded into the Manipulator will be saved. This option is intended to allow the user to quickly clear out the Manipulator and start a new project.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.2. Open Project Menu Option Test

Purpose: To test the "Open Project" menu option to make sure that a previously saved project is loaded into the Manipulator when this option is selected.

Procedure: 1. Prior to choosing the "Open Project" option, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Open Project" menu option under the File menu.
4. Choose a valid SEP project file to be loaded by using the dialog box.

Expected Result: When the "Open Project" option is selected, the user should be prompted to first save any previous information. Then, all of the old information loaded in the Manipulator should be cleared out for a project being loaded and then all previous project information should be reloaded properly.

Comments: This test should probably be completed in conjunction with the next test, which is the "Save Project" option.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.3. Save Project Menu Option Test

Purpose: To test the "Save Project" menu option to make sure that a project is saved properly in the SEP file to be stored for future use.

Procedure: 1. Prior to choosing the "Save Project" option, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Save Project" menu option under the File menu.
4. Choose a valid name and file path for the SEP project file to be created.

Expected Result: When the "Save Project" option is selected, the user should be prompted to choose a location and file name for the SEP project file. Then, all of the current information loaded in the Manipulator should be saved in the project file being created.

Comments: This test should probably be completed in conjunction with the next test, which is the "Open Project" menu option.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.4. Open Picture Menu Option Test

Purpose: To test the "Open Picture" menu option to make sure that a picture and its data are properly loaded into the Manipulator.

Procedure: 1. Have a valid JPEG image and an invalid JPEG image available.
2. Try opening both, one at a time, in the Manipulator

Expected Result: The valid JPEG should be loaded into the Manipulator with all the values loaded into the interface, under the proper headings. The invalid image load attempt should generate an error message about the file structure.

Comments: The Manipulator should not discriminate against file name, but rather the file structure. Even if the file is a valid JPEG but labeled as .BMP or some other format, the file should still load properly.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.1.5. Update Picture Menu Option Test

Purpose: To test the "Update Picture" menu option to make sure that a picture is generated from the values that are currently loaded in the Manipulator interface (whether they are user updated or not).

Procedure: 1. Load a picture into the Manipulator.
2. Before changing any values, try making a replica of the picture by choosing the "Update Picture" menu option.
3. Then try changing some values in the Manipulator and choose the "Update Picture" option again.
4. Using the converted program, convert all 3 files (the original and the 2 new images) and verify that both the files were created with information provided in the Manipulator.

Expected Result: The first picture created should have the exact same values as original converted picture. The second picture should only have values that are different from the original where they were changed/updated in the Manipulator.

Comments: Only change a few values at first to changed to make sure they work properly for all the fields.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.2.1.1.6. Exit Menu Option Test

Purpose: To test the "Exit" menu option to make sure that a the user can properly exit the program.

Procedure:
1. Load a picture into the Manipulator.
2. Change a few values, but don't do anything else.
3. Be sure NOT to save before hitting the "Exit" option.
4. Choose the "Exit" menu option.

Expected Result: Before the application is closed, the user should be prompted to save the current information. Then, after the user has provided input, the application should be closed.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.2 Edit Menu Tests

This section of the test plan is to test all of the Edit Menu options. Each of the Edit Menu options has a test under this section.

3.2.1.2.1. Copy Menu Option Test

Purpose: To test the "Copy" menu option to make sure that when text is selected, we copy it to the system clipboard.

Procedure:
1. Load a picture into the Manipulator.
2. Highlight some data values.

3. Click the “Copy” menu option.
4. In some other program, like notepad or word, try pasting the text in.

Expected Result: The highlighted text in the Manipulator should be pasted to the new document. Also, the text in the Manipulator should remain unchanged.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.2.2. Cut Menu Option Test

Purpose: To test the “Cut” menu option to make sure that the user can cut text out of a given field and paste it back into another.

Procedure:
1. Load a picture into the Manipulator.
2. Highlight some data values.
3. Click the “Cut” menu option.
4. In some other program, like notepad or word, try pasting the text in.

Expected Result: The highlighted text in the Manipulator should be pasted to the new document. Also, the text in the Manipulator should be removed from the text control.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.2.3. Paste Menu Option Test

Purpose: To test the “Paste” menu option to make sure that the user can paste text into the different text controls in the Manipulator.

Procedure: 1. Load a picture into the Manipulator.
2. Highlight some data values.
3. Click the “Copy” menu option.
4. Highlight some other data values.
5. Click the “Paste” menu option.

Expected Result: The highlighted text in the Manipulator should be pasted to the selected text.

Comments: If for some reason the “Copy” menu won’t work, use <ctrl+c> button, which is guaranteed to work.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3. View Menu Tests

This section of the test plan is to test all of the View Menu options. Each of the View Menu options has a test under this section.

3.2.1.3.1. Stretch Large Original Menu Option Test

Purpose: To test the “Stretch Large Original” menu option to make sure that the user can both stretch and view normally the large original picture on the Original Picture tab.

Procedure: 1. Load a picture into the Manipulator.
2. Click on the Original Picture tab.
3. Click the “Stretch Large Original” menu option several times.

Expected Result: The Large Original image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.2. Stretch Large Changed Menu Option Test

Purpose: To test the "Stretch Large Changed" menu option to make sure that the user can both stretch and view normally the large changed picture on the Changed Picture tab.

Procedure: 1. Load a picture into the Manipulator.
2. Click on the Changed Picture tab.
3. Click the "Stretch Large Changed" menu option several times.

Expected Result: The Large Changed image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.3. Stretch Small Original Menu Option Test

Purpose: To test the "Stretch Small Original" menu option to make sure that the user can both stretch and view normally the small original picture on the Console tab.

Procedure: 1. Load a picture into the Manipulator.
2. Click on the Console Picture tab.
3. Click the "Stretch Small Original" menu option several times.

Expected Result: The Small Original image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.4. Stretch Small Changed Menu Option Test

Purpose: To test the "Stretch Small Changed" menu option to make sure that the user can both stretch and view normally the small changed picture on the Console tab.

Procedure:
1. Load a picture into the Manipulator.
2. Click on the Console Picture tab.
3. Click the "Stretch Small Changed" menu option several times.

Expected Result: The Small Changed image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.3.5. Stretch All Menu Option Test

Purpose: To test the "Stretch All" menu option to make sure that the user can both stretch and view normally all of the images in one click.

Procedure:
1. Load a picture into the Manipulator.
2. Click the "Stretch All" menu option several times.
3. Each time you click the "Stretch All" option be sure to check all of the pictures to make sure they updated correctly.

Expected Result: The Small Changed image should toggle between stretch mode and normal mode. Also, when the image is in stretch mode, there should be a check mark next to the menu option.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.1.4. About Menu Tests

This section of the test plan is to test all of the About Menu options. Each of the About Menu options has a test under this section.

3.2.1.4.1. About Menu Option Test

Purpose: To test the "About" menu option to make sure that the user can view the project information and the people associated with the project.

Procedure: 1. Click the "About" menu option.

Expected Result: A new window should open up with the appropriate project information. This window should close when is it clicked on.

Comments: Try several times in a row to make sure it works right.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2. Button Control Tests

This section of the test plan is to list out the menu functions that need to be tested. Included in this section is each of the tests, a short description of the test and the expected results.

3.2.2.1. Project Sub-Tab Button Tests

This section of the test plan is to test all of the buttons on the Project sub-tab. Each of the buttons on the Project sub-tab has a test under this section.

3.2.2.1.1. New Project Button Test

Purpose: To test the "New Project" button to make sure that a new project is created when this option is selected.

Procedure: 1. Prior to clicking the "New Project" button, open a new picture in the Manipulator.
2. Then click the "New Project" button under the Project sub-tab on the Console tab.

Expected Result: All of the old information in the Manipulator should be cleared out for a new project to be created and they should be prompted for a new project file name and path.

Comments: This is not required to make a new project, for instance, you could just load in a picture and then click the "Save Project" button and the current information loaded into the Manipulator will be saved. This option is intended to allow the user to quickly clear out the Manipulator and start a new project.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.2. Load Project Button Test

Purpose: To test the "Load Project" menu option to make sure that a previously saved project is loaded into the Manipulator when this option is selected.

Procedure: 1. Prior to clicking the "Load Project" button, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Load Project" button located under the Project sub-tab under the Console tab.
4. Choose a valid SEP project file to be loaded by using the dialog box.

Expected Result: When the "Load Project" button is clicked, the user should be prompted to first save any previous information. Then, all of the old information loaded in the Manipulator should be cleared out for a project being loaded and then all previous project information should be reloaded properly.

Comments: This test should probably be completed in conjunction with the next test, which is the "Save Project" button.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.3. Save Project Button Test

Purpose: To test the "Save Project" button to make sure that a project is saved properly in the SEP file to be stored for future use.

Procedure: 1. Prior to clicking the "Save Project" button, open a new picture in the Manipulator.
2. Change a few values in some of the text controls.
3. Then click the "Save Project" button under the Project sub-tab under the Console.
4. Choose a valid name and file path for the SEP project file to be created.

Expected Result: When the "Save Project" button is clicked, the user should be prompted to choose a location and file name for the SEP project file. Then, all of the current information loaded in the Manipulator should be saved in the project file being created.

Comments: This test should probably be completed in conjunction with the next test, which is the "Open Project" button.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.4. Load Picture Button Test

Purpose: To test the "Load Picture" button to make sure that a picture and its data are properly loaded into the Manipulator.

Procedure: 1. Have a valid JPEG image and an invalid JPEG image available.
2. Try opening both, one at a time, in the Manipulator by clicking on the "Load Picture" button.

Expected Result: The valid JPEG should be loaded into the Manipulator with all the values loaded into the interface, under the proper headings. The invalid image load attempt should generate an error message about the file structure.

Comments: The Manipulator should not discriminate against file name, but rather the file structure. Even if the file is a valid JPEG but

labeled as .BMP or some other format, the file should still load properly.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.5. Save Picture Button Test

Purpose: To test the "Save Picture" button to make sure that a picture load in the changed picture image boxes are properly saved to file as a JPEG image.

Procedure: 1. Open a valid JPEG image in the Manipulator.
2. Alter a few values and create an image that is not the same, but still viewable as a JPEG image.
3. Click the "Save Picture" button located on the Project sub-tab of the Console tab.

Expected Result: The viewable JPEG loaded into the Manipulator changed image boxes should be saved to file. The values saved should be the values that are currently loaded into the text controls (except the encoded stream) of the Manipulator.

Comments: This image should be tested by trying to open the created JPEG image in a standard image viewer (or multiple viewers for that matter). This image should be viewable as normal.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.1.6. Update Picture Button Test

Purpose: To test the "Update Picture" button to make sure that a picture is generated from the values that are currently loaded in the Manipulator interface (whether they are user updated or not).

Procedure: 1. Load a picture into the Manipulator.
2. Before changing any values, try making a replica of the picture by clicking the "Update Picture" button.
3. Then try changing some values in the Manipulator and click the "Update Picture" button again.
4. Using the converted program, convert all 3 files (the original and the 2 new images) and verify that both the files were created with information provided in the Manipulator.

Expected Result: The first picture created should have the exact same values as original converted picture. The second picture should only have values that are different from the original where they were changed/updated in the Manipulator.

Comments: Only change a few values at first to changed to make sure they work properly for all the fields. You may want to use the converter to convert the images produced hexadecimal to evaluate the data contained in the image.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.2. Huffman and Quantizer Sub-Tab Button Tests

This section of the test plan is to test all of the buttons on the Project sub-tab. Each of the buttons on the Project sub-tab has a test under this section.

3.2.2.2.1. Clear Button Tests

Purpose: To test the all "Clear" buttons on their corresponding text controls on both of the Huffman sub-tabs and the Quantizer sub-tab.

Procedure: 1. Prior to clicking the "clear" button, open a new picture in the Manipulator.
2. Try altering text in each for the Huffman tables and Quantizer tables.
3. Then, for each of the different Quantizer and Huffman table fields, click the corresponding clear button.

Expected Result: The corresponding table field should be cleared out. Also, check to make sure that click one clear doesn't affect any of the other text fields.

Comments: Most images won't have 4 Quantizer and/or 8 Huffman tables, so to test the unused fields, simply type some data into the field and then hit the "Clear" button. Also, if the field hasn't been previously altered, then it original data should be moved to the corresponding original data field.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.2.2. Random Button Tests

Purpose: To test the all "Random" buttons on their corresponding text controls on both of the Huffman sub-tabs and the Quantizer sub-tab.

Procedure: 1. Prior to clicking the "Random" button, open a new picture in the Manipulator.
2. Try clicking the "Random" button to add a random byte onto the end of the tables.

Expected Result: The corresponding table field should have a random byte appended to the end of it. Also, make sure that if the field has not been altered previously, that the information be moved to the corresponding original data TextBox control.

Comments: Most images won't have 4 Quantizer and/or 8 Huffman tables, so to test the unused fields, simply hit the "Random" button, a byte will still be added to the end of an empty table. Also, if the field hasn't been previously altered, then it original data should be moved to the corresponding original data field.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.2.2.3. Restore Button Tests

Purpose: To test the all “Restore” buttons on their corresponding text controls on both of the Huffman sub-tabs and the Quantizer sub-tab.

Procedure: 1. Prior to clicking the “Restore” button, open a new picture in the Manipulator.
2. Change some data values in the Manipulator to get the data input into the corresponding original text field.
3. Try clicking the “Restore” button to restore the originally loaded data into the corresponding tables.

Expected Result: The corresponding table field should be restored to the original value loaded from the image file.

Comments: Most images won’t have 4 Quantizer and/or 8 Huffman tables, so to test the unused fields, simply hit the “Restore” button, the original table will be restored to the corresponding field.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3. General Tests

This section of the test plan is to test all of the other functions not covered by nay other section here. Each of the test in this section reflects some piece of the manipulator that has not previously been tested by any other section in the document.

3.2.3.1. TextBox Control Test

This section of the test plan is to test all of the TextBox controls found within the Manipulator. Each of the tests are described in their following section.

3.2.3.1.1. Changeable TextBox Control Tests

Purpose: To test the all “TextBox” controls in the Manipulator to make sure they are working properly.

Procedure: 1. Open a new picture in the Manipulator.

2. For each TextBox control that is not "grayed out," change some data values. If no data currently exists in the field, then just try adding some text into the control.

Expected Result: The data should be entered into the proper TextBox control, if the control is not "grayed out." If you encounter a control where this doesn't work, please write down the name of each one.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3.1.2. Non-Changeable TextBox Control Tests

Purpose: To test the all non-changeable "TextBox" controls in the Manipulator to make sure they are working properly.

Procedure: 1. Open a new picture in the Manipulator.
2. For each TextBox control that is "grayed out," try to change some data values. If no data currently exists in the field, then just try adding some text into the control.

Expected Result: The data should NOT be entered into the proper TextBox control. If you encounter a control where this doesn't work, please write down the name of each one.

Comments: The non-changeable fields aren't as important as the changeable ones, but they should still all be checked. This will ensure that the original data won't be destroyed, so that the user can restore it if needed.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3.1.3. Generating New JPEG Image Tests

Purpose: To test to make sure that the new image being created includes all of the values currently stored in the Manipulator and only those values.

Procedure:
1. Open a new picture in the Manipulator.
2. Try changing a bunch of different values for the picture.
3. Generate the new picture.
4. Use the converter to convert the original image and the newly generated image to an ASCII file and compare all of the data values.

Expected Result: The only data that should be changed in the newly generated image file from the original file is the data that was updated. Also, this updated data should be reflected in the new file as well.

Comments: The Converter should be sufficient to do this, but you may also want to run the newly generated picture through an image viewer (if the new image itself is viewable). You should use the Design document to evaluate whether or not the format of this file is correct.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.2.3.1.4. Generating New SEP Project File Tests

Purpose: To test to make sure that the new SEP file being created includes all of the values currently stored in the Manipulator and only those values.

Procedure:
1. Open a new picture in the Manipulator.
2. Try changing a bunch of different values for the picture.
3. Generated the new JPEG picture.
4. Add some project notes into the Project Notes text field.
5. Then save the SEP project file.
6. Then open the SEP file in some text processor, like NotePad or Word. You should be able to see all of the values saved in this file.

Expected Result: All of the changed file information should be stored in this file. Also the path and file name of both the original JPEG image and the changed JPEG image should be shown here as well. You should use the Design document to evaluate whether or not the format of this file is correct.

Comments: None.

Date: March 6, 2004

Tester: Geoffrey Griffith

Outcome: Pass

3.3. Web Site Test

This section of the test plan document outlines the series of tests created to test all the functionality of the ISE Website. The web site test will be broken into the following categories:

1. **The Menu Frame Page**
2. **The Main Frame Pages**

The tests and results for both of these categories are compiled in the following sections of this document.

3.3.1. The Menu Frame Page

This section of the Test plan outlines tests done on the page Button.html, which appears in the Menu frame.

3.3.1.1. The Home Button

Purpose: This test is to verify that the Home button links to the correct page.

Procedure: Click the Home button on the Menu.

Expected Result: The page Home.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.2. The Project Proposal Button

Purpose: This test is to verify that the Project Proposal button links to the correct page.

Procedure: Click the Project Proposal button on the Menu.

Expected Result: The document ProjectProposal.pdf should open in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.3. The Documentation Button

Purpose: This test is to verify that the Documentation button links to the correct page.

Procedure: Click the Documentation button on the Menu.

Expected Result: The page DocumentIndex.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.4. The Project Sponsor Button

Purpose: This test is to verify that the Project Sponsor button links to the correct page.

Procedure: Click the Project Sponsor button on the Menu.

Expected Result: The page Sponsor.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.5. The Team Info Button

Purpose: This test is to verify that the Team Info button links to the correct page.

Procedure: Click the Team Info button on the Menu.

Expected Result: The page Team_ISE.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.6. The Download Button

Purpose: This test is to verify that the Download button links to the correct page.

Procedure: Click the Download button on the Menu.

Expected Result: The page Download.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.7. The Links Button

Purpose: This test is to verify that the Links button links to the correct page.

Procedure: Click the Links button on the Menu.

Expected Result: The page Links.html should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.1.8. The Message Board Button

Purpose: This test is to verify that the Message Board button links to the correct page.

Procedure: Click the Message Board button on the Menu.

Expected Result: The page index.php should open in the Main Frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2. The Main Frame Pages

This section outlines the tests done on the various pages displayed in the Main frame.

3.3.2.1. DocumentIndex.html Requirements Button

Purpose:	This test is to verify that the Requirements button on the DocumentIndex.html page functions correctly.
Procedure:	Click the Requirements button on the page.
Expected Result:	A .pdf reader should open ISEFinalRequirements.pdf file in the Main frame.
Comments:	None.
Date:	March 6, 2004
Tester:	Andrew Pouzeshi
Outcome:	Pass

3.3.2.2. DocumentIndex.html Prototype Plan Button

Purpose:	This test is to verify that the Prototype Plan button on the DocumentIndex.html page functions correctly.
Procedure:	Click the Prototype Plan button on the page.
Expected Result:	A .pdf reader should open ISEPrototypePlan.pdf file in the Main frame.
Comments:	None.
Date:	March 6, 2004
Tester:	Andrew Pouzeshi
Outcome:	Pass

3.3.2.3. DocumentIndex.html Sys Arch Design Button

Purpose:	This test is to verify that the Sys Arch Design button on the DocumentIndex.html page functions correctly.

Procedure: Click the Sys Arch Design button on the page.

Expected Result: A .pdf reader should open ISESystemArchitectureDesign.pdf file in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.4. DocumentIndex.html Design Document Button

Purpose: This test is to verify that the Design Document button on the DocumentIndex.html page functions correctly.

Procedure: Click the Design Document button on the page.

Expected Result: A .pdf reader should open DesignSpecFinal.pdf file in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.5. Sponsor.html Project Sponsor Button

Purpose: This test is to verify that the Project Sponsor button on the Sponsor.html page functions correctly.

Procedure: Click the Project Sponsor button on the page.

Expected Result: The page located at http://www.cs.colorado.edu/people/tom_lookabaugh.html should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.6. Download.html Production Code Button

Purpose: This test is to verify that the Production Code button on the Download.html page functions correctly.

Procedure: Click the Production Code button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the zip file code.zip.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.7. Download.html Manipulator Button

Purpose: This test is to verify that the Manipulator button on the Download.html page functions correctly.

Procedure: Click the Manipulator button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the zip file manipulator.zip.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.8. Download.html .NET Framework Button

Purpose: This test is to verify that the .NET Framework button on the Download.html page functions correctly.

Procedure: Click the .NET Framework button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the file dotnetfx.exe.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.9. Download.html Alpha Test Button

Purpose: This test is to verify that the .NET Framework button on the Download.html page functions correctly.

Procedure: Click the .NET Framework button on the page.

Expected Result: The browser should prompt a window asking the user where they would like to download the file dotnetfx.exe.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.10. Links.html www.ijg.org/ Button

Purpose: This test is to verify that the www.ijg.org/ button on the Links.html page functions correctly.

Procedure: Click the www.ijg.org/ button on the page.

Expected Result: The page located at http://www.ijg.org should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.11. Links.html rijndael algo Button

Purpose: This test is to verify that the rijndael algo button on the Links.html page functions correctly.

Procedure: Click the rijndael button on the page.

Expected Result: The page located at http://www.esat.kuleuven.ac.be/~rijmen/rijndael/ should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

3.3.2.12. Links.html Project Sponsor Button

Purpose: This test is to verify that the Project Sponsor button on the Links.html page functions correctly.

Procedure: Click the Project Sponsor button on the page.

Expected Result: The page located at http://www.cs.colorado.edu/people/tom_lookabaugh.html should be displayed in the Main frame.

Comments: None.

Date: March 6, 2004

Tester: Andrew Pouzeshi

Outcome: Pass

4. SUMMARY

This document gives a detailed test plan for the ISE Production Code, Manipulator, and the ISE web site. These tests should be sufficient to prove that the Production Code, Manipulator, and web site are functioning correctly. All of the results uncovered by the team members during the alpha testing process have been listed here as well.

5. RELATED READINGS

[Chang and Li 96]

Chang, H. and Li, X. *On the Application of Image Decomposition to Image Compression and Encryption*. 1996.

Describes image degradation based on compression and encryption.

[Chang and Li 2000]

Chang, H. and Li, X. *Partial Encryption of Compressed Images and Videos*. 2000.

Describes a partial encryption scheme used on compressed multimedia files.

[Droogenbroek and Benedett 2002]

Droogenbroek, M. and Benedett, R. *Techniques for Selective Encryption of Uncompressed and Compressed Images*. 2002.

[Kailasanathan and Naini 2003]

Kailasanathan, C. and Naini, R. *Compression Performance of JPEG Encryption Scheme*. 2003.

Describes compression performance of JPEG encryption.

[Daigaku and Griffith and Jarchow and Kadhim and Pouzeshi]

Daigaku, S., Griffith, G., Jarchow, J., Kadhim, J. and Pouzeshi A. *Requirement Specification.* 2003.

Describes the requirement for Team ISE and for the ISE project.

[Daigaku and Griffith and Jarchow and Kadhim and Pouzeshi]

Daigaku, S., Griffith, G., Jarchow, J., Kadhim, J. and Pouzeshi A. *System Architecture.* 2003.

Describes the high-level system architecture for the ISE project.

[Li and Knipe and Cheng 97]

Li, X., Knipe, J. and Cheng, H. *Image Compression and Encryption Using Tree Structures*. 1997.

Describes compression methods that utilize tree structures.

[Lookabaugh and Sicker and Keaton and Guoand and Vedula 2003]
Lookabaugh, T., Sicker, D., Keaton, D., Guoand, W. and Vedula, I. *Security Analysis of Selectively Encrypted MPEG-e Streams*. 2003.

Description of the methods and results of applying selective encryption to MPEG-2 streams.

[Miano 99]
Miano, J. *Compressed Image File Formats*. Addison Wesley Longman, Inc., Reading, Massachusetts, 1999.

Provides a description of the JPEG file format.

[Norcen and Uhl 2003]
Norcen, R. and Uhl, A. *Selective Encryption of the JPEG2000 Bitstream*. 2003.

Describes a selective encryption scheme on JPEG2000 files.

[Pennebaker and Mitchell 93]
Pennebaker, W. and Mitchell J. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, New York, 1993,

Provides a thorough description of the JPEG file format and its components.

[Podesser and Schmidt and Uhl 2002]
Podesser, M., Schmidt, H. and Uhl, A. *Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments*. 2002.

Describes Bitplane Encryption.

[Seo and Kim and Yoo and Dey and Agrawal 2003]
Seo, Y., Kim, D., Yoo, J., Dey, S., Agrawal, A. *Wavelet Domain Imag Encryption by Subband Selection and Data Bit Selection*. 2003.

Describes Wavelet Domain and Data Bit encryption methods.