**Steganography Project Design**

Linden Crandall, Jonathan Mainhart, Zhihua Zheng

University of Maryland Global Campus

CMIS 495: Current Trends and Projects in Computer Science

Prof. Majid Shaalan

April 12, 2022

Table of Contents

[Table of Figures 3](#_Toc100155832)

[Project Overview 4](#_Toc100155833)

[Main Goals 4](#_Toc100155834)

[Typical User Interaction 4](#_Toc100155835)

[User Interface 5](#_Toc100155836)

[GUI Overview 5](#_Toc100155837)

[Main Goals 5](#_Toc100155838)

[File Access 5](#_Toc100155839)

[Data Structures 5](#_Toc100155840)

[Input and Output 6](#_Toc100155841)

[Class, Method, and Function Names 6](#_Toc100155842)

[File I/O 7](#_Toc100155843)

[Message Encoding 7](#_Toc100155844)

[Message Decoding 7](#_Toc100155845)

# Table of Figures

[Figure 1. Logic Control Flow of a typical user interaction. 5](#_Toc100155592)

# Project Overview

## Main Goals

Provide a GUI application which runs locally on any computer which meets the following minimum system requirements:

* Python 3.9 or greater installed
* A keyboard or similar text input device
* A mouse or similar pointing device
* A monitor or similar display capable of rendering images

Allow a single user to open an image file and determine whether a secret text message has been encoded in the pixel data.

Allow a user to enter a text message into a text input area and encode that text into the pixel data of an image.

Allow a user to save an encoded image file with the same or different filename.

Alert the user to any problems reading or writing files from within the application.

## Typical User Interaction

Figure 1 shows a typical user interaction with the application.

First, the user starts the program and chooses an image via the file selection interface. The image is then displayed to the user in the window.

Next, the image pixel data is then automatically extracted and sent to the message decoding algorithm. If a message can be decoded, it will be displayed inside the text input area of the window, otherwise a default message will be displayed.

Next, the maximum allowed character limit will be calculated and displayed to the user as a ratio of characters entered to characters allowed.

The user may then delete, edit, or append the text within the text input area until the character limit is reached. When the user is ready, they can push the “encode” button.

When the “encode” button is pressed, the user’s input will be converted to its binary value. The binary value of the message will then be used to encode the message in the RGB value of the image’s pixels.

Upon completion of the encoding process, the user may then choose to reset the pixel data to its original state, or to save the image for later use. The user will be alerted if they attempt to overwrite a file.

If the image is reset, the original message (if any) will be displayed in the text input area.

Diagram

Description automatically generated

Figure . Logic Control Flow of a typical user interaction.

# User Interface

## GUI Overview

The Graphical User Interface (GUI) of the application will have an image display area and four function buttons on the right side of the image. The function buttons are “Open Image”, “Encode Image”, “Reset Image” and “Save Image”. The “Reset Image” button will be initially disabled until an image is successfully encoded. There will be a text field at the bottom of the GUI to enable the user to enter the secret data to be encoded and to read messages that have been decoded from images. Between the text field and the image display area will be a label to display warning messages regarding the text field (e.g., The image has been reset!).

## Main Goals

Enable the user to interact with the application using graphical elements presented in a logical layout.

## File Access

The GUI will require read access to image files on the user’s system to provide visual feedback to the user.

## Data Structures

Images will be .jpg, .jpeg, or.png

Image pixel data will be stored as a List of Tuples

User input will UTF-8 characters stored as a String object

## Input and Output

User input will be passed to the encoding function as a String type.

Image pixel data will be returned from the \_\_ function as a List of Tuples.

Image pixel data will be passed to the encoding function as a List of Tuples.

Encoded image pixel data will be returned from the encoding function as a List of Tuples.

Encoded image pixel data will be passed to the file\_write function as a List of Tuples.

## Class, Method, and Function Names

**Table #**

gui.py – MainFrame class

|  |  |  |
| --- | --- | --- |
| **Method** | **Input** | **Output** |
| on\_open\_button\_click | self | filename |
| on\_encode\_button\_click | self | encode\_image(rgb\_pixel\_data, gui\_text\_input) |
| on\_reset\_button\_click | self | reset\_image() |
| on\_save\_button\_click | self | save\_image(encoded\_rgb\_pixel\_data) |

**Table #**

stego.kv -MainFrame

|  |  |  |  |
| --- | --- | --- | --- |
| **Element Name** | **Type** | **Properites** | **Action** |
| Image | Image | source: default or filename returned from on\_open\_button\_click() | N/A |
| Open Image | Button | background-color: # 4CAF50  size: ?  position: ?  enabled by default: True | on\_open\_button\_click() |
| Encode Image | Button | background-color: # 4CAF50  size: ?  position: ?  enabled by default: False | on\_encode\_button\_click()  enables when file opened |
| Reset Image | Button | background-color: # 4CAF50  size: ?  position: ?  enabled by default: False | on\_reset\_button\_click()  enables when message encoded |
| Save Image | Button | background-color: # 4CAF50  size: ?  position: ?  enabled by default: False | on\_save\_button\_click()  enables when file opened |
| gui\_label | label | default text: (0/0) No Image Loaded  text: (n/p) characters remaining  text: Maximum (p) characters entered | N/A |
| gui\_text\_input | TextInput | default text: decoded message from loaded image  text: user input text | N/A |

# File I/O

File I/O design.

# Message Encoding

## Overview

Messages will be encoded into the target image by first extracting the image pixel data into a list of tuples which represent the red, green, and blue (RGB) color values of the image. Next, each character in the user message string will be converted to its 8-bit binary ASCII value. The 8-bit values will then be used to shift the RGB values to be either an even number to represent a binary zero, or an odd number to represent a binary one. Pixels will be read 3 at a time which will allow one 8-bit character to be encoded. The remaining bit of each set will tell the decoder to either continue reading (binary one), or to stop (binary zero).

## Main Goal

main goals

## File Access

No file access is required.

## Data Structures

String: User input

List: Binary converted user string

List: RGB pixel data

## Input and Output

The required input and output for the encode process functions are described in Table #.

**Table #**

Encode Process Functions

|  |  |  |
| --- | --- | --- |
| Function | Input Parameters | Return Value |
| encode\_secret\_message | String: user\_input, list: pixel\_data | list: pixel data |
| convert\_message | String: user\_input | list: converted\_message(08b) |
|  |  |  |

# Message Decoding

Message decoding design.