COMP 8505

Assignment 2 Design Document

Image Steganography

By: Derek Wong

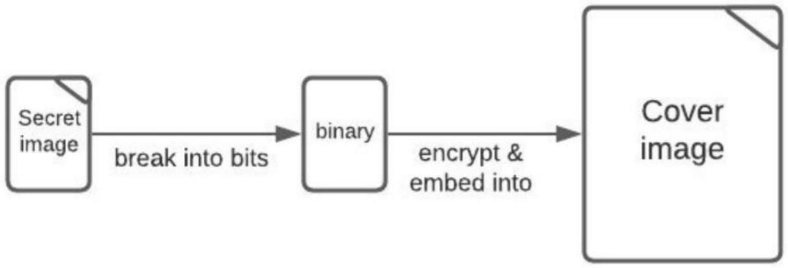
Instructor: D’Arcy Smith

Due: October 22nd 2300 hrs, 2021

Table of Contents

Overview

To become familiar with Steganography and to design and implement an application that allows users to encrypt a secret image and embed it into a cover image. The output image should visually look the same as the original cover image. The simple diagram below illustrates the conversion process.



Requirements

The application will be a command-line application with appropriate switches to perform the various functions. The two main functions will be the embedding (hide) and the extraction functions. The filenames of the cover image, secret image, and output file will also be specified as part of the command line invocation of the program.

Constraints

1. Image format for the application will be BMP
2. Use a helper function that will display all the switches and command line arguments for the application
3. Implement encryption (DES)
4. Use 8 bytes in the cover image to hide a single byte of the secret image. Other data that will also need to be hidden is the filename, including the extension. The file sizes will be calculated from both images, to ensure that the cover file is enough to hold all of the secret image data.

Approach

The application will be developed in python. Therefore, it should work on both Linux and Windows (since python is a cross-platform language). The pixels hiding operations will be implemented using a python library called "**Pillow**". The library provides APIs for reading pixels from images, editing the pixels, creating images from raw pixels. Other python modules such as "**argparse** " and "**ntpath** " are used for improving the utility of the application.

Finally, the encryption functionality will be implemented with the symmetric-key algorithm "DES". The user input password will be factored into a key of length 8, 16, or 24 bytes, and used for encrypting /decrypting the pixels and filename of the secret image.

Application Design

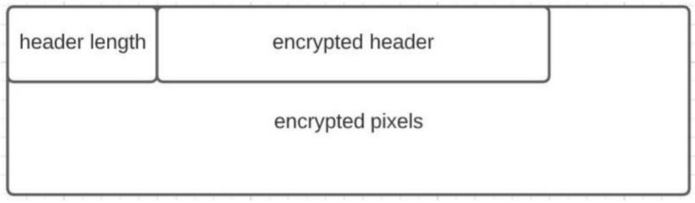
Embedding mechanism

Pixels hiding

Each pixel has the values R, G and B. Each value can be represented in binary (For instance, 11111111 represents hex value 255). The idea is to hide a single bit in the last bit of a RGB value. Each pixel of the secret image can be split and hidden into 9 pixels of the cover image.

Headers

To know the filename and size of the secret image, a header length (integer value) and encrypted header is embedded before the pixel data of the secret image. Below figure shows the structure of header (before embedding into pixels of cover image)



The application will first read the header length (stop when encounter space character). Retrieve header length of bytes after, then decrypt it to acquire header information. The header information includes the filename and dimension of the secret image.

Modules

The application consists of three modules - dcstego.py, dcutils.py and dcimage.py. Source codes are separated into these modules based on type of functionality.

- **dcstego.py** - Application entry point, includes functionality such as parsing command line arguments, checking file size and file formats. Responsible for invoking methods for data encryption/extraction and images loading provided in other modules.
- **dcutils.py** - Includes core functions for hiding and extracting data from cover images. This module also includes simple functions for data type conversion (bits to ascii, int to bits... etc)
- **dcimage.py** - Has a single class that acts as a wrapper of the python Pillow module. Includes core functionality such as loading images, providing next pixels in line, editing pixels at a specific point or returning properties of an image.

Pseudo code

dcstego.py

```
# application entry point
main() {
   parse user argument
   If user actions is create {
      check if input password valid, exit if invalid
      proceed to stego image creation
   else
      check if input password valid, exit if invalid
      Proceed to stego image extraction
}
```

dcutils.py

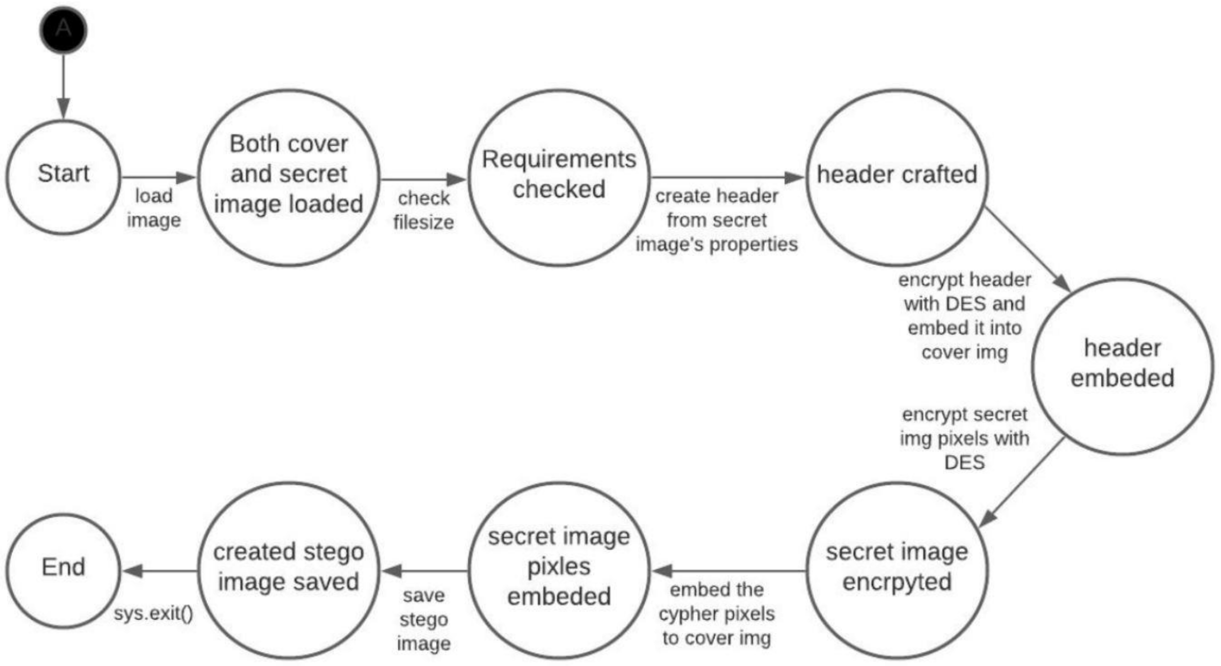
```
# methods of data conversion
ascii to bits (char)
bits to ascii (bits arry)
int to bits (int)

# methods for hiding bits in pixels 3 + 3 + 2 = 8 (1byte)
hide 3 bits (pixels to hide into, 3 bits to hide)
hide 2 bits (pixels to hide into, 2 bits to hide)
hide 1 byte in 9 pixels (byte to hide, pixels to hide into)

# methods for data extraction, image loader can return next pixel in line
extract 1 byte from 9 pixels (pixels to extract from)
extract first int from a series of pixels (image loader instance, delimiter)
extract a fixed length of character from pixels (image loader instance, delimiter)
```

dcimage.py

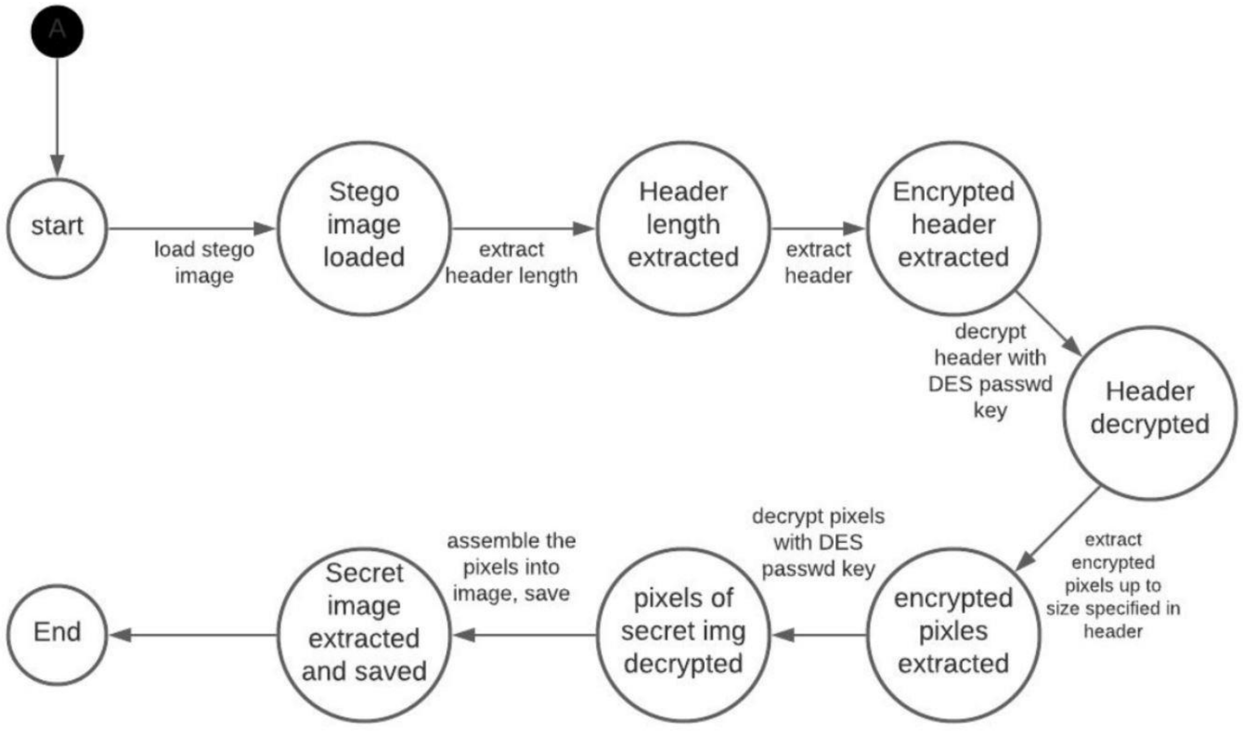
```
class PixelLoader {
   # variables
   properties of the image loaded (dimension, width height etc...)
   coordinates for the next pixel to return

   # constructor
   constructor(filename of the image to load)

   # class methods
   getter and setters for image properties ()
   return next pixel in line ()
   edit pixel at specific coordinate (coordinates)
   save image as separate image file (filename)
   convert all pixels of the load image to int array () # used for encryption/decryption
}
```

State diagram

Encryption process


A
Start
load image
Both cover and secret image loaded
check filesize
Requirements checked
create header from secret image's properties
header crafted
encrypt header with DES and embed it into cover img
header embeded
encrypt secret img pixels with DES
secret image encrpyted
embed the cypher pixels to cover img
secret image pixles embeded
save stego image
created stego image saved
sys.exit()
End

Decryption process


A
start
load stego image
Stego image loaded
extract header length
Header length extracted
extract header
Encrypted header extracted
decrypt header with DES passwd key
Header decrypted
extract encrypted pixels up to size specified in header
encrypted pixles extracted
decrypt pixels with DES passwd key
pixels of secret img decrypted
assemble the pixels into image, save
Secret image extracted and saved
End