F28HS2 Hardware Software Interface
C Coursework

This coursework involves building a simple steganography system in C. This system will have a command-line interface (CLI) that allows its users to encode/hide a message into an image, then reveal/decode the text from the image. Both encoding and decoding will use secret keys. The application can be used by Alice and Bob who shared a secret, say “2560867”. Alice can use the “encode” function to hide “Hello” into an image using the secret “2560867”. Then, Alice can send the modified image to Bob, who can use the “decode” function with the same key “2560867” to extract the hidden message.

We hide text in images in this coursework by replacing the least significant bits in some selected pixels of the image with bits from the characters we want to hide. The pixels are chosen in a particular order that the receiver can reproduce. The text is then retrieved by re-generating the pixel sequence, in the same original order, and extracting message bits from them.

Least Significant Bit Technique.

Modifying the least significant bit (*LSB*) of each byte of a pixel causes minor modifications that is not easy to spot by the human eye. For example, figure 1 shows one pixel whose initial RGB values are:

- R: 00 00 10 10
- G: 01 01 00 00
- B: 10 10 00 01

By overwriting the least significant bits (0 in the R byte, 0 in the G byte and 1 in B byte) with three bits “100”, we produce a new pixel that encodes

- 1 in the LSB of the R byte
- 0 in the LSB of the G byte (this byte has not been modified)
- 0 in the LSB of the B byte

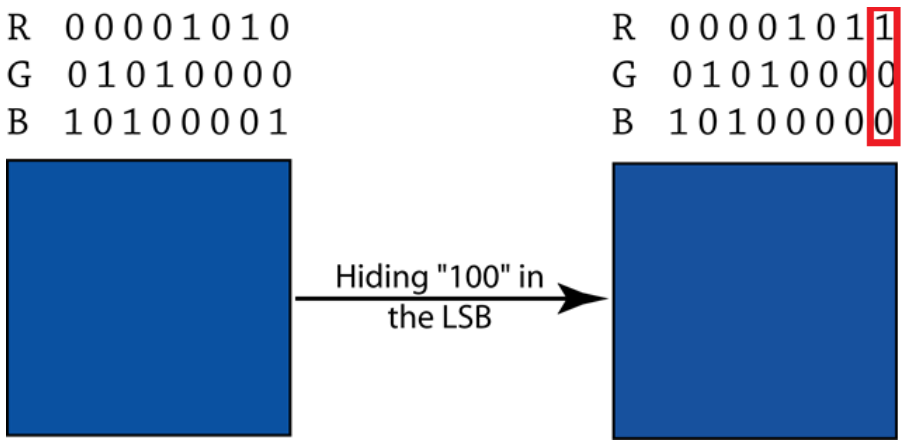

Figure 1: Least Significant Bit Technique

The difference between the two pixels is minor.

PPM Format.

You will work with PPM format images. PPM is a very simple open source RGB colour bitmap format. PPM files' format is:

P3
 # comment1
...
 # commentN
width height
max
r1 g1 b1
r2 g2 b2
r3 g3 b3
...

where:

- *P3* – 2 letter code for PPM format. It is a constant string set to "P3".
- *width* – integer number of columns
- *height* – integer number of rows
- *max* – integer maximum colour value – usually 255
- r_i g_i b_i – integers between 0 and *max* for pixel *i*'s red, green and blue values
- The number of comments is not pre-defined

A) Design a struct PPM to hold PPM images as defined above.

B) Write functions:

- *struct PPM * getPPM(FILE * f)* to return a C structure PPM image from file *"f"*;

- *showPPM(struct PPM * im)* to display the PPM image *"im"* as text in the above format;

- *struct PPM * encode(struct PPM * im, char * message, unsigned int mSize, unsigned int secret)* to return a modified copy of PPM image *"im"* with *"message"* characters hidden in its pixels using *"secret"*; *"mSize"* is the size of *"message"*.
  Note.
  - The least significant bits of each of the three bytes of a pixel are used to hide 3 bits from the message. The message is hidden in the image by iteratively choosing a pixel, then overwriting the LSBs of the pixel with the next 3 bits to hide.
  - Pixels are chosen in a particular order that has to be reproducible. That is, if the sender encoded the message in, say, pixels 5, 30 and 7, then the receiver needs to be able to generate the same sequence (in

the same order) before they start extracting the message bits from the LSBs of the pixels. The pixel order should depends on the secret shared by both parties. In other word, to be able to reproduce the pixel order one must know the algorithm and the secret.
    - You will need to decide on how to choose the next pixel; options include the rand function from C, or more sophisticated functions (e.g., MD4, MD5, SHA1, secure pseudo random generators, etc.).

- *char * decode(struct ppm * im, unsigned int secret)* to return the message hidden in the pixels of PPM image *"im"* using *secret*;

C) Write a program *steg* to encode and decode messages in PPM images.

- For the call:

$ steg e file1.ppm > file2.ppm

steg prompts for a message and a secret and shows *file1.ppm* with the message encoded within it. Here, the output is redirected to *file2.ppm*.

- For the call:

$ steg d file.ppm

steg prompts for the secret, then shows the message hidden in *file.ppm* if the secret is the right one (or a wrong message, extracted from the image using the wrong secret).

Possible extensions.

The following are extra, challenging, tasks you might want to do to increase your chances of getting an A. We will still need to check the basic encode and decode functions are working: make sure you make these as extra versions of your basic encode and decode functions.

- Consider hiding your message in more than the 3 least significant bits. Make this a parameter n that the user can set to any value between 1 and 21. Check the effect of this on the quality of the image.
- Create an extra version that hides an image inside another one. Make it hide the most significant b bits of each byte the secret message inside the least significant bits of each byte of the container image. Make b vary between 2 and 7.
- Use a Cryptographically Secure Pseudo-Random Generator (CSPRG) to generate the next pixel's position.
- Integrate OpenSSL (the cryptography library) in your code. Offer the user several options for generating pixels positions.

- Using OpenSSL, offer the possibility to encrypt (e.g., using AES256) the message before hiding it in the image.
- 32-bit (*sizeof(unsigned int)*) secrets are considered to be weak (90-bit keys are crackable nowadays). Offer the possibility to use a string secret rather than the unsinged integer one.

Plagiarism

We uploaded a set of slides, entitled "Academic Misconduct" with more details on plagiarism and other forms of academic misconduct as well as a few tips on how to avoid them. You can find the slides in the same folder as this description. Notable things to take into consideration:

- Coursework reports must be written in your own words and any code in your coursework must be your own code. If some text or code in the coursework has been taken from other sources, these sources must be properly referenced.
- Failure to reference work that has been obtained from other sources or to copy the words and/or code of another student is plagiarism and if detected, this will be reported to the School's Discipline Committee. If a student is found guilty of plagiarism, the penalty could involve voiding the course.
- Students must never give hard or soft copies of their coursework reports or code to another student. Students must always refuse any request from another student for a copy of their report and/or code.
- Sharing a coursework report and/or code with another student is collusion, and if detected, this will be reported to the School's Discipline Committee. If found guilty of collusion, the penalty could involve voiding the course.
- And remember: the consequences of taking unacceptable short cuts in coursework are much worse than getting a bad mark (or even no marks) on a piece of coursework. There has been one case this year where a student was awarded on Ordinary degree (rather than an Honours degree) because of the sanction imposed by the University's Discipline Committee. The offence was plagiarism of coursework.

Further information on academic misconduct can be found in: https://www.hw.ac.uk/students/doc/discguidelines.pdf
Your coursework submissions will be automatically checked for plagiarism.

Submission.

You should submit your:

- program design, outlining your choice of data structures and algorithms;
- program listing;

by Thursday 5th March at 3:30pm.

Subsequently, you will be required to demonstrate your work during the following week, on unknown PPM files which we will provide. The exact time of the demo session will be announce on the due course.

Notes.

1. There's a description of PPM at http://netpbm.sourceforge.net/doc/ppm.html
2. You can view PPM files with gimp on Linux or ImageViewer on Raspbian.
3. You can use od to look at what's inside a PPM file. For example:

```
$ od –x –a ape.ppm | more
```

4. The C random number library is in stdlib. Calling srand(int) will seed the random number generator with the int. Subsequent calls to rand() will return a number in the range 0 to RAND_MAX. So you can get a value in the range 1 to N from rand()%N+1.
5. Your PPM representation should keep all the comments.
6. Your program should be able to deal with PPM files with arbitrary numbers of rows and columns.