project[10]: cryptographic images

Due Tuesday, 5/3/2016, 12:59:59pm

Project Goals

The goals of this project are to:

1. Build upon some prior projects
2. Utilize functions, multidimensional arrays, bitwise operators, and file I/O
3. Learn more about images
4. Implement a complex algorithm
5. Learn about cryptography and ethics

Important Notes:

1. **Formatting:** Make sure that you follow the precise recommendations for the output content and formatting. Your assignment will be auto-graded and any change in formatting will result in a reduced grade.

2. **Comments:** Header comments are required on all files and recommended for the rest of the course. Points will be deducted if no header comments are included.

Background

You are to write programs to find and hide the steganographic files (http://en.wikipedia.org/wiki/Steganography) in the included set of images. For example, a file is hidden this image:

=> I hope someone gets my message in a bottle.

Here's how we did it. We have two files:

1. the mask image (left, color/grayscale)
2. the hidden file (any format, but I suggest using a text file)

The method for hiding data in an existing image is rather simple. A standard grayscale image is made up of a 2-dimensional array of pixels. Each pixel is just an 8-bit unsigned char (a number between 0 and 255). 8 bits is usually enough to encode most information for a grayscale image. However, while your eye can probably tell the difference between a pixel value of 5 and a pixel value of 200, you probably will not notice a small difference (say the difference between 200 and 201). We can take advantage of this to hide information that a computer can detect but a person could not.

So, if we start with an 8-bit pixel, we can throw out the least significant bit, the one that distinguishes the value of 200 from 201. The binary value of 200 is: 11100010 and 201 is: 11100011.

If we just toss out whatever value is in that last bit, we can hide a single bit from the file we want to hide. (like 1 if the hidden bit should be a 1, and 0 if the hidden bit should be a 0).

More specifically, you need to:

1. Write a program to hide a file inside of another image
    a. allocate memory for a properly-sized multi-dimensional array
    b. for each pixel in the hidden image, encode that information along with the mask image
    c. write the new image to a file
2. Write a program to unmask the hidden file from an encoded image and save it as a decoded file
    a. for each pixel in the encoded image, pull out the least significant bit and use those bits to build bytes
    b. write the bytes to the decoded file

Your goal (for the encode portion): to write a program that will:

1. read three command-line arguments, the first is the image we wish to hide a file in, the second is the filename of the file we wish to hide, the third is the filename that we will write the hidden image to.
2. copy the original image to an array
3. create new pgm file to hold changed image
4. allocate memory for the new image data
5. start at the mask image's 0,0 pixel
6. for each bit in the hidden file
    a. drop the least significant bit from the mask image current pixel value
    b. add a 1 to that pixel value if the corresponding hidden file bit is a 1
    c. add a 0 (or don't) to that pixel value if the corresponding hidden file bit should be a 0
    d. move to the next pixel in the image (the next column in the image, or the next row's 0th column if at the max column has been reached)
7. write an EOF character in the next 8 image pixels
8. save the array to a file using the pgm image format we've been using for the prior projects

Your goal (for the decode portion): to write a program that will:

1. read two command-line arguments; take the first command-line argument which is the filename of an image (we will provide two pgm images to test with)
2. read that image from the provided filename to an array
3. open the decoded file with the second command-line argument as the filename
4. for each pixel in the image
    a. if the last bit is zero
        i. make the next corresponding bit in the hidden file a 0
    b. if the last bit is one

        i. make the next corresponding bit in the hidden file a 1
5. write each decoded byte to the decoded file

If you recall from before, to do this you need to:

- Name your files encode.c and decode.c

Example images can be found here: (http://www.cse.unr.edu/~dave/cs135/)

Challenge 1:
For encode.c, test the files beforehand to make sure that every bit of the code file can fit into the mask image. Save this challenge as encode_challenge.c

Challenge 2:
Create functions:

- One that takes a filename as an argument, and reads a PGM image from a file and return a dynamically allocated 2-D array that stores the image
- One that takes a filename, a dynamically allocated 2-D array, and image dimensions as arguments and writes the image to a file

Save those functions in a separate filename, image.c. Put prototypes for those functions in a header file image.h, and use those files in a revised version of your program, encode_challenge2.c and decode_challenge2.c

Submission details

The project needs to be submitted by Tuesday, 5/3/2016, 12:59:59pm.

To submit your project, you will have to save your project files to an ECC machine using the Linux VM or the nomachine client:

- create a directory called "project10"
- save your *.c files in that directory
- save your description file into that directory
- DO THIS ONCE: Install the submission script ***(don't type the '>' symbols)***
  ```
  > cd ~
  > wget http://www.cse.unr.edu/~newellz2/submit
  > chmod +x ./submit
  ```
- TO Submit:
  ```
  > cd project10
  > ~/submit
  ```

The submission script copies all files in the current directory to our directory. You may submit as many times as you like before the deadline, we only keep the last submission.

Academic Honesty

Academic dishonesty is against university as well as the system community standards. Academic dishonesty includes, but is not limited to, the following:

Plagiarism: defined as submitting the language, ideas, thoughts or work of another as one's own; or assisting in the act of plagiarism by allowing one's work to be used in this fashion.

Cheating: defined as (1) obtaining or providing unauthorized information during an examination through verbal, visual or unauthorized use of books, notes, text and other materials; (2) obtaining or providing information concerning all or part of an examination prior to that examination; (3) taking an examination for another student, or arranging for another person to take an exam in one's place; (4) altering or changing test answers after submittal for grading, grades after grades have been awarded, or other academic records once these are official.

Cheating, plagiarism or otherwise obtaining grades under false pretenses” constitute academic dishonesty according to the code of this university. Academic dishonesty will not be tolerated and penalties can include canceling a student’s enrollment without a grade, giving an F for the course, or for the assignment. For more details, see the University of Nevada, Reno General Catalog.