CS 558 Introduction to Computer Security, Spring 2024
Programming Assignment 2

Out: 3/13/2024 Wed.
Due: 3/23/2024 Sat. 23:59:59

For this programming task, your objective is to develop a program capable of extracting hidden information from a BMP image file

1. Initial State: The first 100 bytes of the BMP file do not contain any hidden (steganographic) information and should be ignored.

2. Identification of Stego Information:

   - Data Extraction: The steganographic data is embedded in the least significant bit (LSB) of each byte in the file, starting after the initial 100 bytes. The less significant bits of each hidden bytes comes first in the bits sequence. To help you understand this, here's an example: suppose the next 8 bits sequence that you read from the file are(in the order you get them) b_0:0, b_1:1, b_2:1, b_3:0, b_4:1, b_5:1, b_6:0, b_7:1. Based on this bits sequence, the reconstructed byte is shown in the Figure1. The reconstructed byte value is `0xb6` in hexadecimal and 182 in decimal.

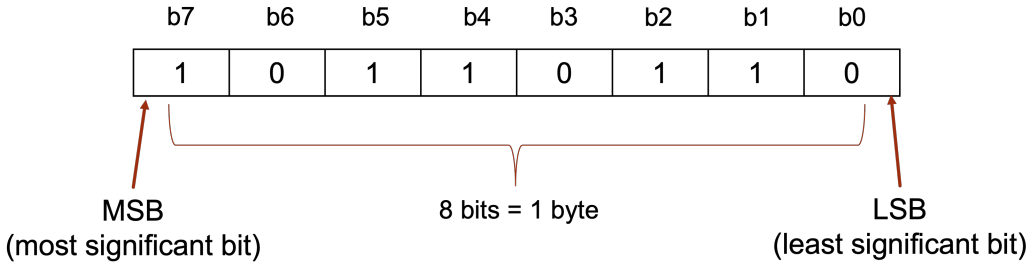

Figure 1: Reconstructed byte from bits sequence

   - Indicator bits: Immediately following the initial 100 bytes, search for an 64 bits sequence in the hidden information where each interpreted byte is the hexadecimal value 0xa5. This sequence serves as a marker indicating the presence of steganographic content.
   - Output Size: Directly after the indicator bits, the next 27 bits in the hidden information represent the size of the extracted information, in bytes. This size does not include the initial 100 bytes, the 64 bits indicator, or the 27 bits used for the size itself.

3. Extraction Process: Continue extracting the least significant bits from subsequent bytes, up to the number of bytes specified by the decoded size value. Collect the extracted bits and compile them into bytes to reconstruct the hidden data. The output should be saved to a file, the name/path of which will be specified via the command line.

Your program must accept the following inputs from the command-line: The name of the input file and the name of the output file. A command-line example:

```
./your_prog input_file output_file
```

Your program should handle any unexpected input correctly, e.g., invalid inputs, non-stegan files. A sample input file is provided. The reconstructed output file from the sample input Alice BMP file is a PDF file of Alice's Adventures in Wonderland. You can use it to verify the correctness of your program.
You can use any programming languages that you are comfortable with for this assignment. Make sure the department machines(Lab G7) or remote machines support your language choices. The assignment will be graded on these machines. Contact the TA if your language is not supported by the department machines.

Log and submit your work

Log your work: besides the source files of your assignment, you must also include a README file which minimally contains your name, B-number, programming language you used and how to compile/execute your program. Additionally, it can contain the following:

- The status of your program (especially, if not fully complete).
- A description of how your code works, if that is not completely clear by reading the code (note that this should not be necessary, ideally your code should be self-documenting).
- Possibly a log of test cases which work and which don't work.
- Any other material you believe is relevant to the grading of your project.

Provide a seperate Makefile if you can, so we can just run a make command instead of manually compile your programming using the instructions in your README which could lead to problems.

Compress the files: compress your README file, all the source files in the same folder, and any additional files you add into a ZIP file. Name the ZIP file based on your BU email ID. For example, if your BU email is "abc@binghamton.edu", then the zip file should be "proj1_abc.zip".

Submission: submit the ZIP file to Brightspace before the deadline.

Grading guidelines

(1) Prepare the ZIP file on a Linux machine. If your zip file cannot be uncompressed, 5 points off.

(2) If the submitted ZIP file/source code files included in the ZIP file are not named as specified above (so that it causes problems for TA's grading scripts), 10 points off.

(3) If the submitted code does not compile or execute:

```
1   TA will try to fix the problem (for no more than 3 minutes);
2   if (problem solved)
3     1%-10% points off (based on how complex the fix is, TA's discretion);
4   else
```

```
5     TA may contact the student by email or schedule a demo to fix the problem;
6     if (problem solved)
7       11%-20% points off (based on how complex the fix is, TA's discretion);
8     else
9       All points off;
```

So in the case that TA contacts you to fix a problem, please respond to TA's email promptly or show up at the demo appointment on time; otherwise the line 9 above will be effective.

(4) If the code is not working as required in this spec, the TA should take points based on the assigned full points of the task and the actual problem.

(5) Late Penalty: Day1 10%, Day2 20%, Day3 40%, Day4 60%, Day5 80%

(6) Lastly but not the least, stick to the collaboration policy stated in the syllabus: you may discuss with your fellow students, but code should absolutely be kept private.