Course	COMP 8505
Program	Bachelor of Science in Applied Computer Science
Term	Jan 2026

- This is an individual programming assignment.

Objective

- This assignment focuses on implementing a steganography tool based on the Least Significant Bit (LSB) method using the PNG format.
- Additionally, you will integrate encryption to secure the embedded data.
- The tool must support encoding (hiding data) and decoding (retrieving data) from PNG images.

Learning Outcomes

- Understand and implement the LSB method for data hiding in PNG images.
- Apply encryption and decryption techniques to secure hidden data.
- Develop a comprehensive solution that integrates encoding, decoding, encryption, and decryption functionalities.
- Enhance problem-solving and programming skills by working with binary data and cryptography.

Assignment Details

You will build a command-line application in a language of your choice (e.g., C, Python, or C++) that performs the following tasks:

Encoding

- Reads a file (e.g. text, image, audio) containing the message to be hidden.
- Encrypts the message using a symmetric encryption algorithm.
- Embeds the encrypted data into the least significant bits of the pixel values in a PNG image.

Decoding

- Extracts the encrypted data from the PNG image.
- Decrypts the extracted data to retrieve the original message.

- Outputs the decoded message to the console or saves it to a file.

Requirements

Functional Requirements

Encoder

- A PNG image file to act as the carrier (cover) image.
- A file containing the message to encode into the image.
- A key for encryption (provided via the command line or as a configuration file).

Decoder

- A PNG image file with the encoded message (stego image).
- A key for encryption (provided via the command line or as a configuration file).
- The decoded and decrypted message after processing the stego image.

Features

- Ability to encode a message into a PNG image and retrieve it accurately.
- Securely encrypt and decrypt the message using a symmetric encryption algorithm.

File Integrity

- Ensure the encoded image remains visually identical to the original, with no noticeable artifacts.
- Preserve the PNG file's structure and metadata where possible.

Encryption Requirements

- Implement or use a pre-existing library to encrypt the hidden message.
- Use a user-supplied encryption key.
- The encrypted message must be embedded in the image, and its decryption should restore the original content.

Steganography Requirements

- To hide data in the pixel values of the PNG image, use the least significant bit (LSB) method.
- The program must handle:
    - RGB images.
    - RGBA images, if applicable, without disrupting the transparency (alpha channel).
- Validate and ensure the message aligns with the carrier's image based on size and colour depth.

Error Handling Requirements

Validate input files

- Ensure the image is a valid PNG file.
- Verify that the input text file is readable and not empty.
- Confirm that the message size does not exceed the image's capacity.

Handle edge cases

- Provide clear error messages if the input is invalid or insufficient.
- Terminate gracefully on unexpected errors (e.g., missing files, unreadable input).

Testing Requirements

Thoroughly test and document your program using the following scenarios:

- Carrier Images:
    - An all-white image.
    - An all-black image.
    - A simple image (e.g., with solid colours or geometric patterns).
    - A complex image (e.g., a photograph with many details).
- Hidden Files:
    - A small text file (e.g., a few sentences).
    - A larger text file (e.g., several paragraphs or a page).
    - A binary file (e.g., an executable or an image).
- Comparison:
    - For each test case, include:
        - The original carrier image.
        - The stego image contains the hidden file.
        - Verification that the stego image visually resembles the original image.
        - The `diff` between the decoded file and the encoded original file to confirm accuracy.

Constraints

- You can use any programming language of your choice.
- The program must support only PNG images.
- The program should not modify any metadata in the PNG file.
- The encoded image must remain visually indistinguishable from the original.
- Third-party libraries for working with PNG files and encryption/decryption (e.g., libpng for C or Pillow for Python) are allowed, but you must write all steganography logic.

Resources

- C: libpng documentation
- Python: Pillow library documentation
- C: OpenSSL documentation
- Python: PyCryptodome documentation

Submission

- Ensure your submission meets all the guidelines, including formatting, file type, and submission.
- Follow the AI usage guidelines.
- Be aware of the late submission policy to avoid losing marks.
- ***Note: Please strictly adhere to the submission requirements to ensure you don't lose any marks.***

Evaluation

Topic	Value
Design	25%
Implementation	25%
Testing	25%
Documentation	25%
Total	100%

Hints

- Use the least significant bits (LSBs) of each pixel's colour values to hide the message.
- Consider using the alpha channel for RGBA images if it does not affect transparency.
- Test your encryption and decryption logic with simple examples before integrating it with the image encoding.
- Ensure the message size does not exceed the image's capacity. A simple calculation is: (Number of Pixels × Bits per Pixel) ÷ 8 ≥ Message Size in Bytes.
- Use small images and short text files for testing.
- Verify that the decrypted message matches the original.