ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA THÀNH PHỐ HỒ CHÍ MINH

Cryptography And Network Security (Lab)

Assignment 1

Lecturer: Nguyễn Đức Thái

Members: Nguyễn Hoàng Long - 1752324

Cù Tấn Phát - 1752408

Nguyễn Ngọc Anh Tuấn - 1752592

Table of contents

I. Steganography algorithm

In this assignment, our group has used python and Pillow and numpy library to process the image

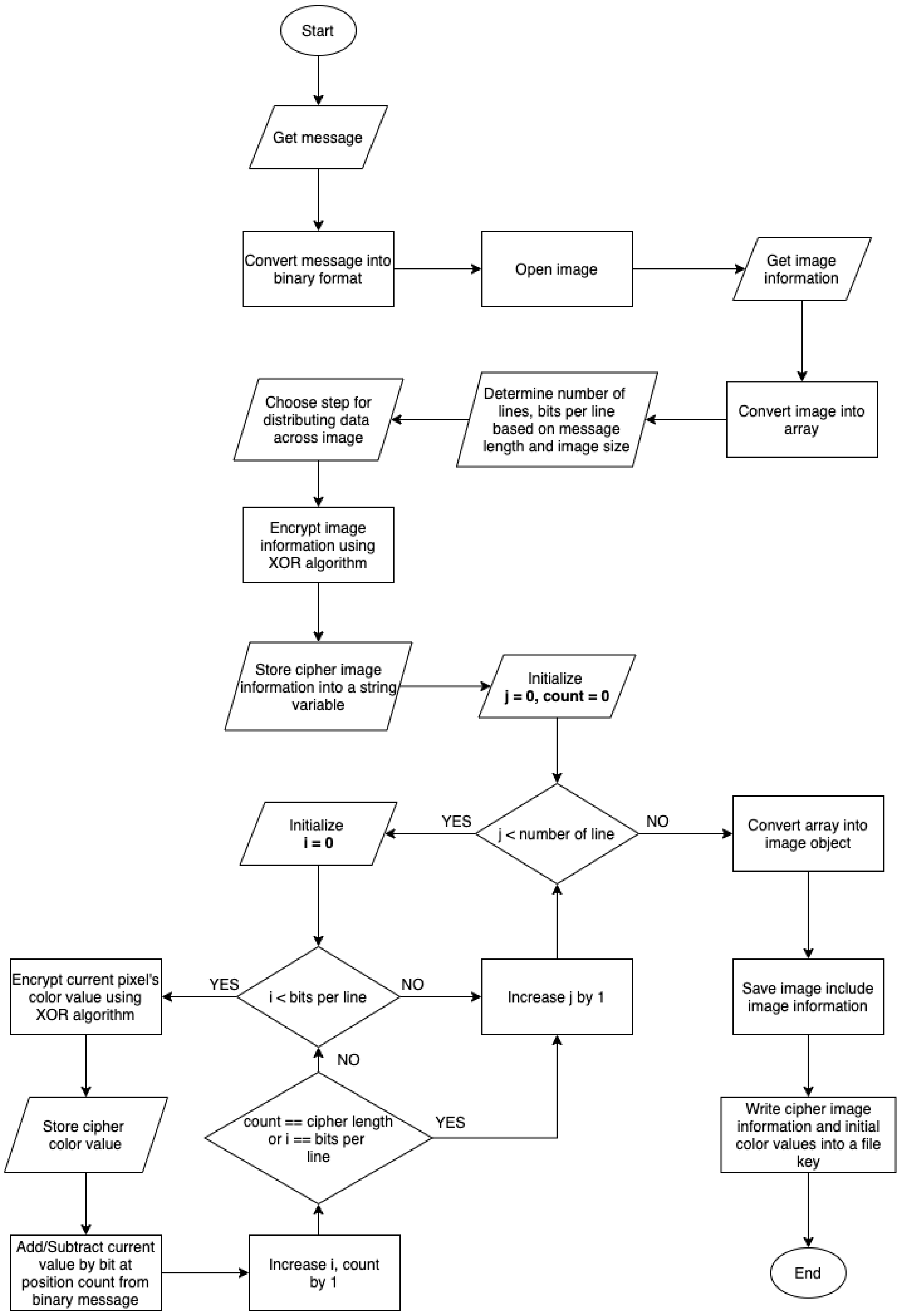
1/ Encryption

a. Description

- At the beginning, we handled the plain message by converting it into the binary format. Each character in the message has been converted into a block of 7 bits including '-'.
- Then, we open the original image to get its current external information such as color profile, exif. To make it easy to process, we had converted the image object into an array of pixel values using numpy library.
- We wanted to distribute the message across the image instead of putting the whole message at a specific area like the head or the tail of the image. As a result, we choose 2 numbers as the step for 2 axis - horizontal and vertical axis.
- To inject the message into the image array, we use 2 loops, one for horizontal and another for vertical. Firstly, we store the current value - the original one, into an array for decryption later. To minimize the change in the image color, we decide that we would add or subtract (in case the result is over 255) by the value of the bit in the message. Because the bit value could only be 0 or 1, it would not change the value of the pixel so much.
- We would add every bit from the message into the image, so our algorithm would run until the last bit is put into the image or we may say when the count variable is equal to the length of the message.
- Then, we convert the numpy array into the image and save it as a PNG image.
- Finally, we store some information (number of lines, number of bits per line,...) as the key into a text file for decryption. To

increase the complexity of our solution, we also encrypted the mentioned information into the binary format before writing them into the file key.

b. Flowchart

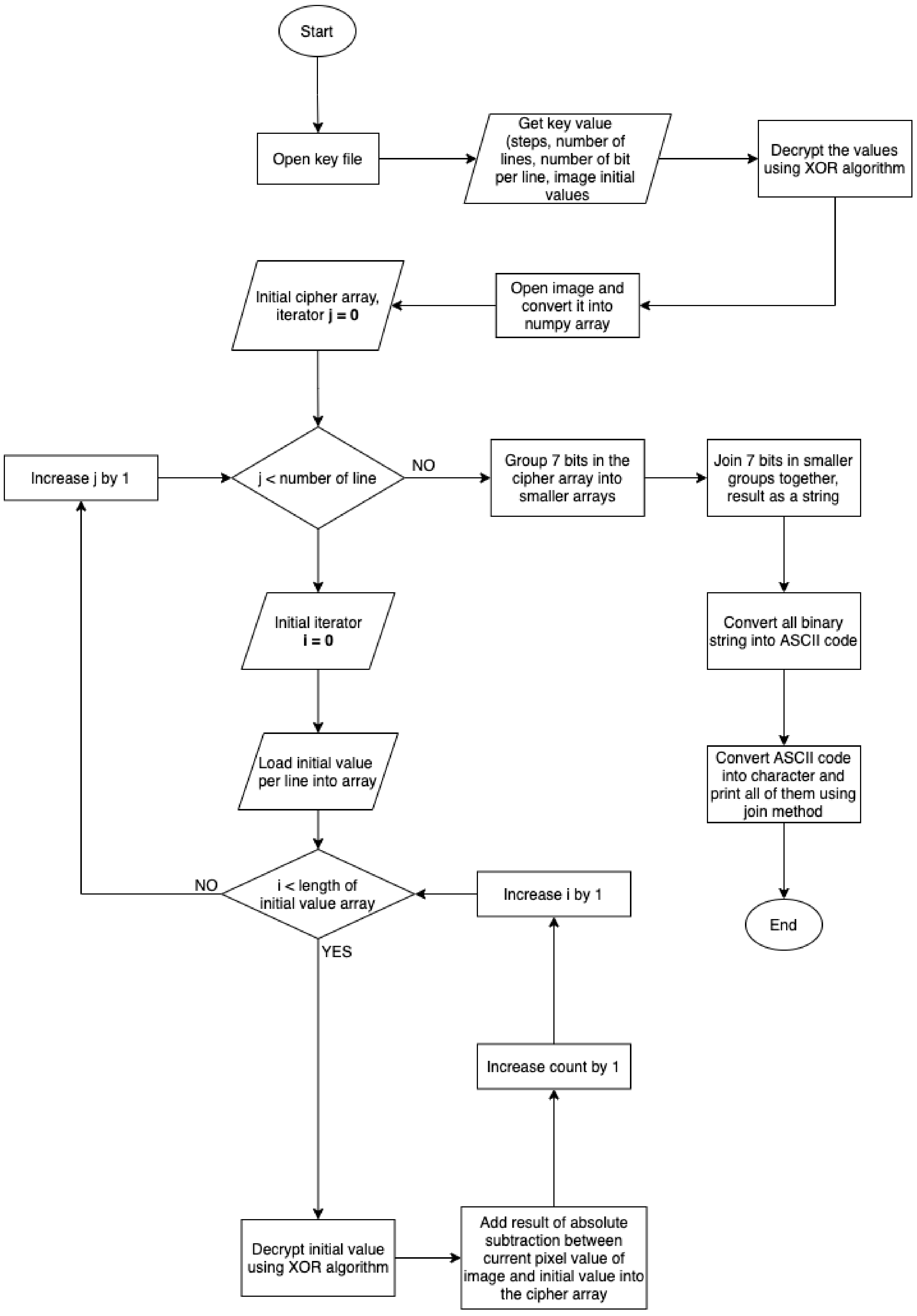

Start
Get message
Convert message into binary format
Open image
Get image information
Convert image into array
Determine number of lines, bits per line based on message length and image size
Choose step for distributing data across image
Encrypt image information using XOR algorithm
Store cipher image information into a string variable
Initialize j = 0, count = 0
j < number of line
YES
NO
Initialize i = 0
Convert array into image object
i < bits per line
YES
NO
Encrypt current pixel's color value using XOR algorithm
Increase j by 1
Save image include image information
NO
count == cipher length or i == bits per line
YES
Store cipher color value
Write cipher image information and initial color values into a file key
Add/Subtract current value by bit at position count from binary message
Increase i, count by 1
End

2/ Decryption

a. Description

- To get the message from the image, we need some information as a key to find the position of message bits which have been inserted into the image. Therefore, we need to read the key file text and load all information from it. Because we have encrypted these information to make bad guys difficult to understand our key, we need to decrypt them before continuing our algorithm.
- Then, we open the image and convert it into the numpy array
- We loop through the number of lines and number of initial values per line which were extracted from the key to find every bit that has been injected into the image. The loop would run until it reaches the end origin.
- To find out the message bit value, we simply take the absolute result of the subtraction between current and initial of the exact pixel which was found based on the step. The result would be only binary value (0 or 1) because we have changed the pixel value by exact bit value in the encryption.
- After the loop, when we get all the message bit values from the image, we would group each 7 bits together and convert them back to the ASCII code.
- Finally, we convert the ASCII code into the character before concatenating all characters and printing them in the terminal.

b. Flowchart



Start
Open key file
Get key value (steps, number of lines, number of bit per line, image initial values
Decrypt the values using XOR algorithm
Open image and convert it into numpy array
Initial cipher array, iterator **j = 0**
j < number of line
NO
Group 7 bits in the cipher array into smaller arrays
Join 7 bits in smaller groups together, result as a string
Convert all binary string into ASCII code
Convert ASCII code into character and print all of them using join method
End
Increase j by 1
Initial iterator **i = 0**
Load initial value per line into array
i < length of initial value array
NO
YES
Increase i by 1
Increase count by 1
Decrypt initial value using XOR algorithm
Add result of absolute subtraction between current pixel value of image and initial value into the cipher array

II. Complexity

It is extremely difficult for a bad guy to find out the message that we have inserted into the image because of below points:

a. Algorithm
   - With our algorithm, the new image is almost unchanged after we inserted the message into it. Some pixels are exactly the same with their own value in the original image. As a result, it is impossible to find the difference between two images using normal eyes.
   - Some bad guys may use tools to check the difference between 2 images to find out. However, some pixels keep their original value, so the bad guys could not get all the bits that were inserted into the image.
   - In our algorithm, we had generated a file key for decryption but in the format of binary string and stored information had been encrypted too. As a result, if a bad guy somehow got the key, it would still be difficult for him to understand what that file key supposed to do

b. Key
   - As we mentioned, after the encryption, there would be a file key which would be used to support the decryption algorithm to get the message from the image out.
   - Without the key file, the decryption algorithm would not be able to work and the bad guys could not get the message from the image in spite of using checking difference tools between two images because of the unchange of some bits value