School of Science and Technologies

RMIT UNIVERSITY

OENG1207 – Digital Fundaments

LABORATORY EXPERIMENT 2 REPORT

Ciphers

GROUP INFORMATION

Group 2, Team 10

Tuesday, 13:00 – 15:00

STUDENTS

Nguyen Quoc Hoang	s3697305
Nguyen Tan Huy	s3864185
Phan Ngoc Quang Anh	s3810148

SUBMISSION DUE DATE

August 23, 2020

I. Part 2 - Encrypting/Decrypting text

Task 1 - Problem Statements

In this part, we will be designing and implementing an algorithm to encrypt and decrypt a text message based on the Exclusive OR operator. Whether it is a photo, a video or a music file, computer stores all types of information as a combination of on and off, or binary values 1 and 0 in the registers. A bit is the smallest unit of computer data that represents the on or off state of a register. By manipulating the bits the way we design, we can create a cipher text to protect sensitive information and it is only meant to be read with a provided key.

There are files for this part, one will be encrypting a plain text and the other will be doing the reverse by decrypting the cipher text. The inputs/outputs for each function will be as followed:

1. Encoder:
   - Input data: plain text that is read in from a text file.
   - Output data: cipher text that will be written to a new text file and a cipher key.
2. Decoder:
   - Input data: a cipher key and cipher text obtained from the encoding process
   - Output data: decrypted text

Since this lab only deals with character arrays, we are making the assumption that only plain text contained in the ".txt" files are allowed to solve the problem.

Task 2 - Algorithm Design

We created 2 user-defined functions, called "XOR_encoder" and "XOR_decoder" to handle the encryption and decryption processes. The heart of this algorithm is to convert a text message in the form of character array and a cipher key to binary form and perform the exclusive OR operator on those arrays of binary. We then

went back up from binary to obtain new characters. The algorithm design for these 2 functions is illustrated in the following flowchart:

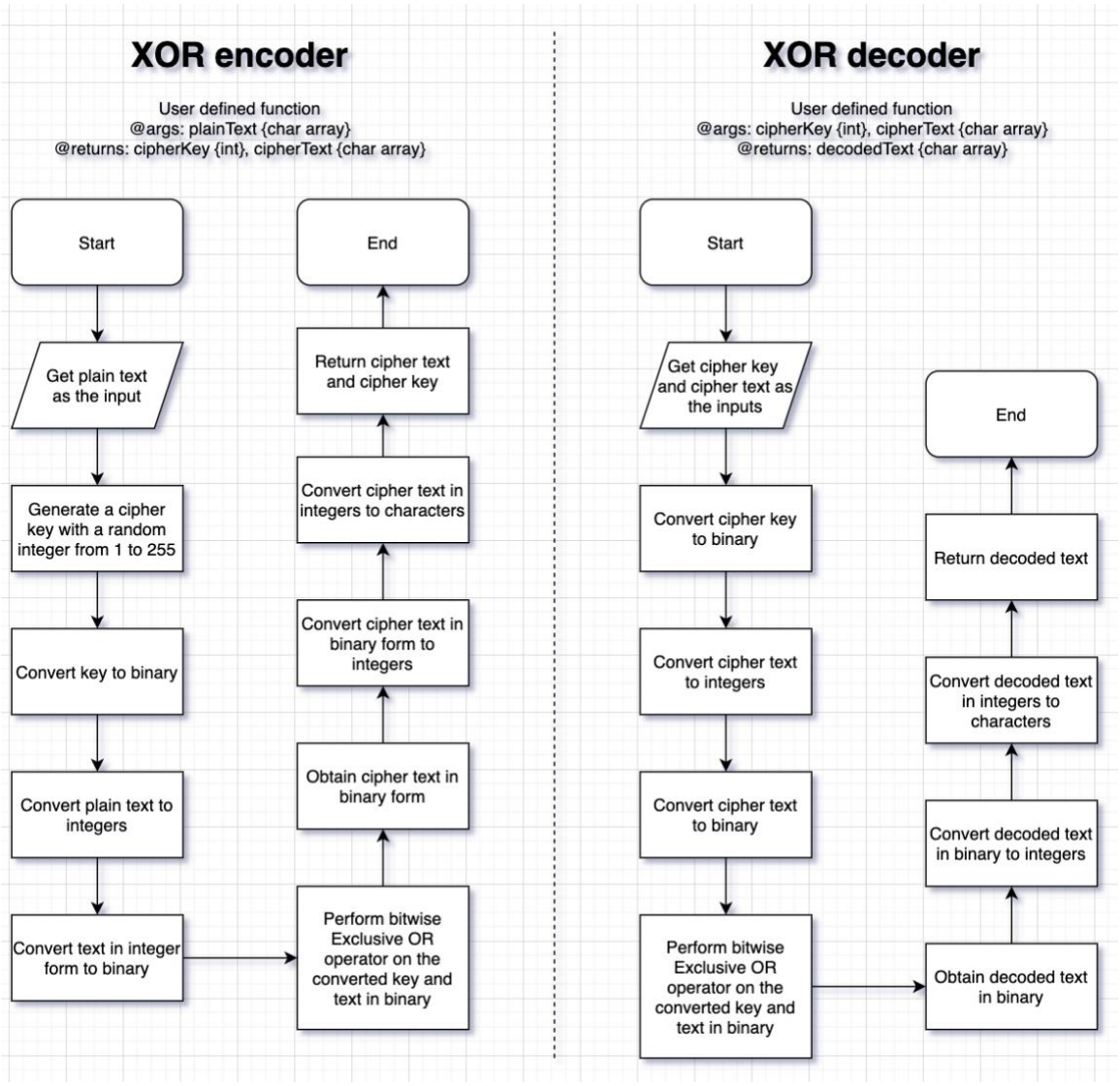

Figure 1 - XOR encoder/decoder algorithm design

To obtain the message, we created another script called “Part2.m” that handles 2 processes: reading plain text from a file and writing encrypted and decrypted message to new files. This script will be calling both functions “XOR_encoder” and “XOR_decoder” above to perform the encrypting/decrypting. The algorithm design for this file is as followed:

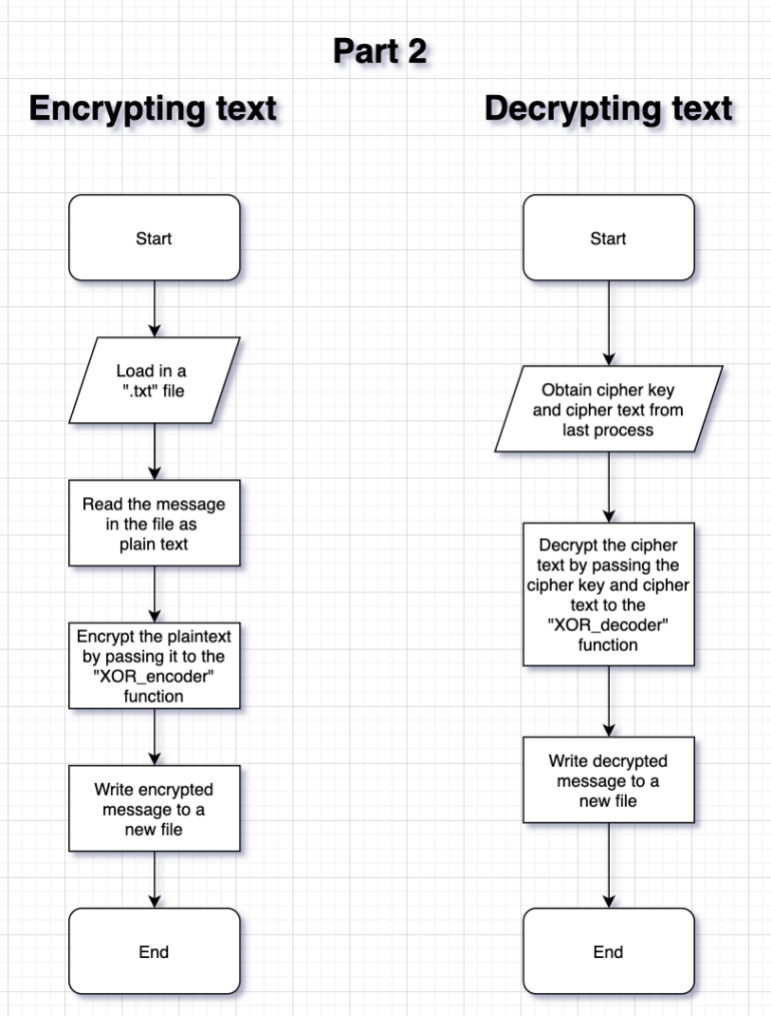

Figure 2 - Part 2 Algorithm design

We will be using ‘abc’ as our plain text and number 7 as our key. Convert ‘abc’ to integer, we get the character array [97, 98, 99] as these are the ASCII values of those letters. Convert the text and the key to binary and we get [01100001, 01100010, 01100011] and [00000111] respectively. Perform an XOR operator on these 2 values and we get the cipher text 'f ¦ &’.

Task 3 – MATLAB Encoder/Decoder Program

The MATLAB script for this part can be run from file “Part2.m”. It will call 2 other files for encrypting and decrypting text. Running this file will require a user to load in a text file, and it will output 3 files: “filename_encrypted.txt”, “filename_decrypted.txt” and “filename_forPart3.txt” that contains only the

encrypted message that will be used for part 3. To ensure the program works correctly, the message in the original file and in the "filename_decrypted.txt" should be the same.

II. Part 3 – A Brute-Force Decryption Algorithm

Task 4 – Problem Statements

In this problem we are required to create a function that can perform a brute-force attack on an encrypted message. To our understanding, using the term brute-force is a figurative way of referring to a problem-solving method which focuses on sheer power rather than intellectual strategies. In this particular task, the brute-force solution will examine every possible case to reach a desired conclusion which means that our function is going to decipher the message by trying all of the encryption keys one by one to get the result.

For the input we have a random encrypted message which is an array of characters.

For the output we will produce a text file that contains all the possible combinations between the cipher key and the cipher text. The decrypted message should be somewhere in the file.

Task 5 – Algorithm Design

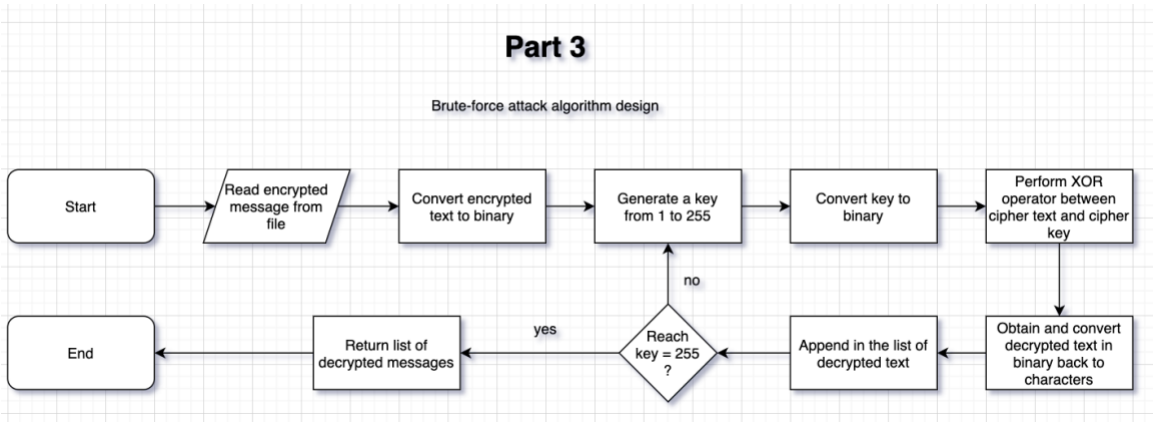

Figure 3 - Brute-force Algorithm design

The flow chart above shows the design for our brute-force algorithm. Basically, it is another XOR decoder, the difference is that we don't know the encryption key. Therefore, we have to test every cipher text with every possible key in 255 extended ASCII values. We will obtain a text file containing a list of 255 sets of decrypted messages, and the result will possibly be in the file.

Task 6 – MATLAB Brute-force attack program

The MATLAT program to perform brute-force attack on an encrypted message is contained in the "Part3.m" script. Note that obtain the cleanest result, please select the file generated from Part 2 with the suffix "_forPart3.m" as it only contains the encrypted message. The result will be written in to a new filed called "_bruteforced.txt".

III. Conclusion

The project revolves around a very interesting subject which is about ciphers, encryption and decryption methods thus giving knowledge on how ciphers are generated and how to decode them. Most importantly, this project allows us to have a better understanding of an important feature of MATLAB: user-defined functions, as well as data-converting techniques. It also helps us review what we have learned in MATLAB programming like iterative statements, the ASCII table, etc. All of these are valuable experience enabling us to be competent users of MATLAB programming as well as such a high applicable field like encryption and security.