EHN 410 PRACTICAL 2 INSTRUCTIONS

AES Functionality:

Your program must allow me to input any text I want to be encrypted along with the password in ASCII format. Your program should display the output of the *Subbytes*, *shifting rows, mixing the columns, XOR* between the state and the expanded key and finally the entire *expansion keys.* The expansion key and the expanded message should be displayed at the end of all output. **All output should be in Hex format.**

AES CBC

You should make a dedicated directory called "CBC output" to store the encrypted and decrypted output files. I will test this with various files to check that your code can decrypt and encrypt properly. Your program must allow me to select any file I want to be encrypted/decrypted via the terminal commands. Your program must take input to select the desired file, the mode (cbc 128, cbc 192, cbc 256), the IV in **ASCII format**, the key in ASCII format, and finally a tag to either encrypt or decrypt a file.

AES CFB

You should make a dedicated directory called "CFB output" to store the encrypted and decrypted output files. Your program must allow me to select any file I want to be encrypted/decrypted via the terminal commands. Your program must take input to select the desired file, the mode (cfb 128, cfb 192, cfb 256), the IV in **ASCII format**, the key in ASCII format, the stream length, and finally a tag to either encrypt or decrypt a file.

Readme.md

This file must contain all the usage instructions on how to compile and run your code. It must contain the commands needed to encrypt/decrypt using CBC or CFB.

Parameters:

- -e encryption
- -d decryption
- -cbc <len> cbc encryption/decryption
- -cfb <len> cfb encryption/decryption
- <len> either 128, 192 or 256
- -t <text to decrypt>
- -key <password>
- -iv <initialization vector>
- -fi <input file>
- -fo <output file>
- -streamlen <len> length of the stream (for cfb: either 8, 64 or 128)
- -h help

Example usage:

1. -e -cbc 128 -fi <input file> -fo <encrypted file> -key <password> -iv <initialization vector>
2. -d -cbc 192 -fi <encrypted file> -fo <decrypted file> -key <password> -iv <initialization vector>
2. -d -cfb 192 -fi <encrypted file> -fo <decrypted file> -key <password> -iv <initialization vector>
3. -e -cfb 256 -t <text> -key <password> -iv <initialization vector> -streamlen <len>

ALL INPUT TO THE TERMINAL MUST BE STRUCTURED IN THE SAME WAY AS IN EXAMPLE USAGE. AND IT MUST FOLLOW THE EXACT ORDER AS IN EXAMPLE USAGE

Your program must be able to encrypt/decrypt pdf, jpg, png and txt files.
Your code must be properly commented.
Your doxygen should contain class/function diagrams.