BBM 465: Information Security Laboratory
Programming Assignment-1

Teachers:
- Ahmet Selman Bozkır
- Ali Baran Taşdemir

Students:
- Mert Tazeoğlu (21946606)
- Mehmet Emin Yıldız (21527602)

Problem Definition

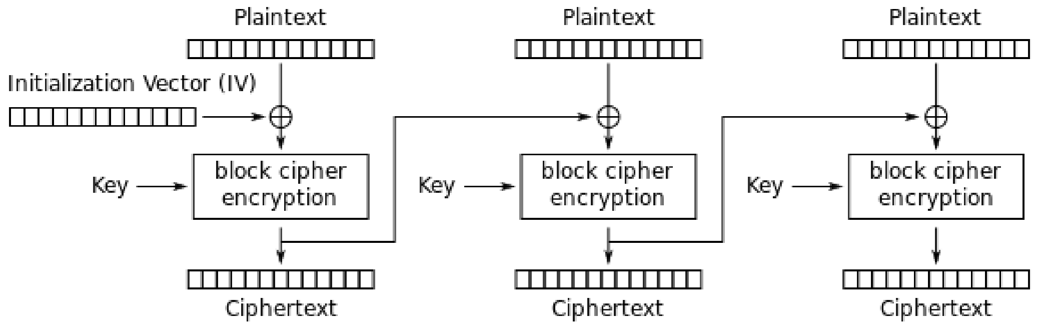
In this assignment we implemented a encryption / decryption tool which is named FileCipher.

Implementation and Testing Details

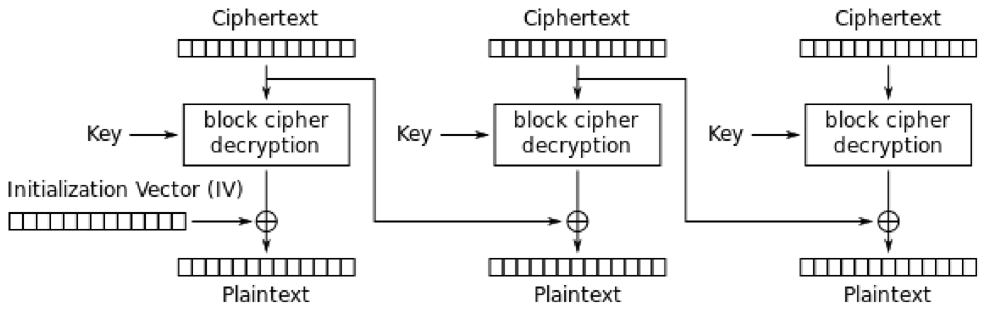
1. Encryption Modes and Encryption Algorithms

Our program supports four different encryption modes and two different encryption algorithms. We implemented encryption modes and used Java's Crypto API in order to apply DES and 3DES algorithms to texts that encrypted with encryption modes.

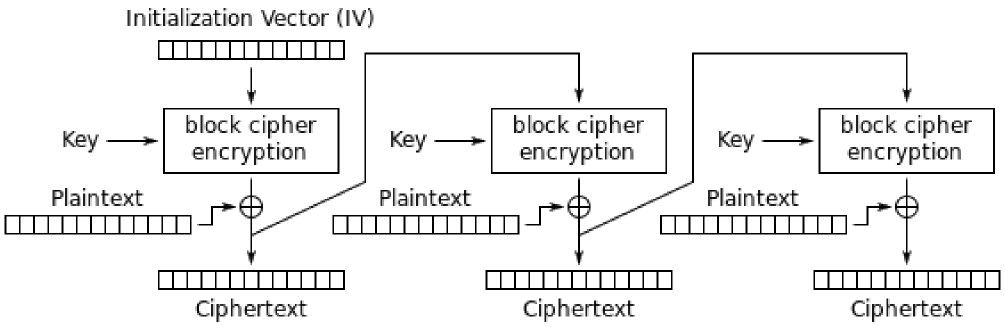
Program takes an input file about encryption and another input file about keys. At the first, program reads these files and stores required variables such as keys. Then, with using these keys program encrypts input text file. In encryption modes, program reads each line input file and uses each line as plaintext.



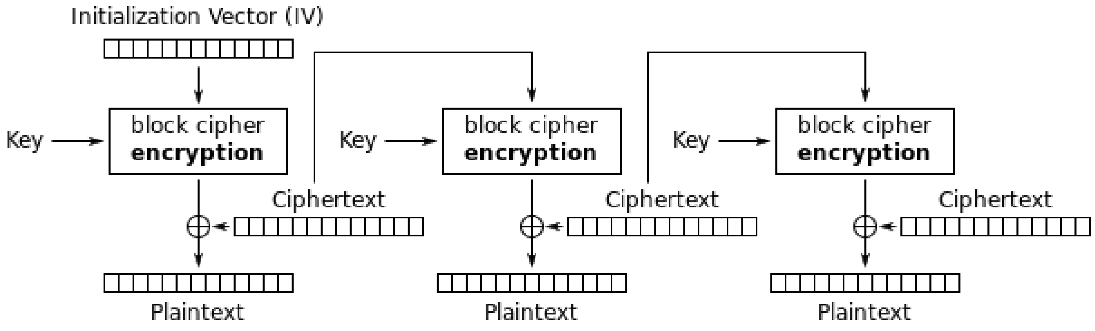
Cipher Block Chaining (CBC) mode encryption



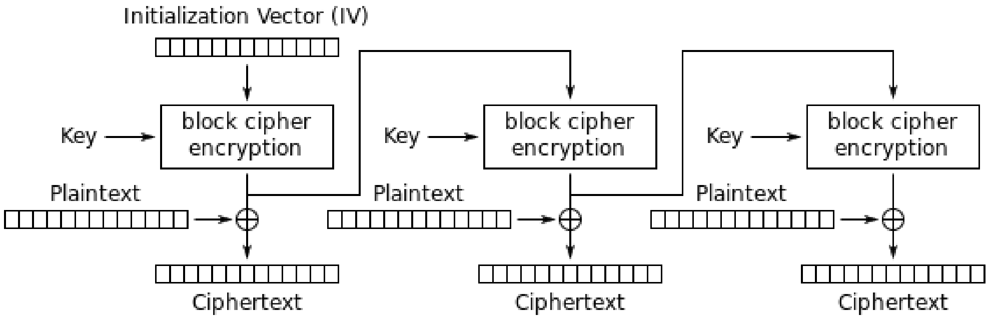
Cipher Block Chaining (CBC) mode decryption



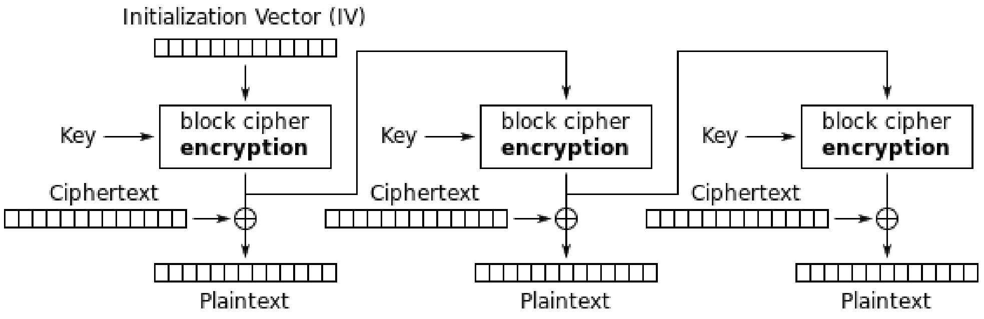
Cipher Feedback (CFB) mode encryption



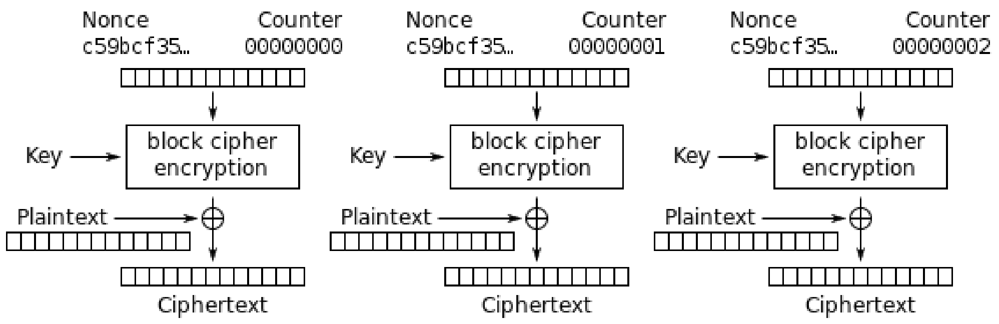
Cipher Feedback (CFB) mode decryption



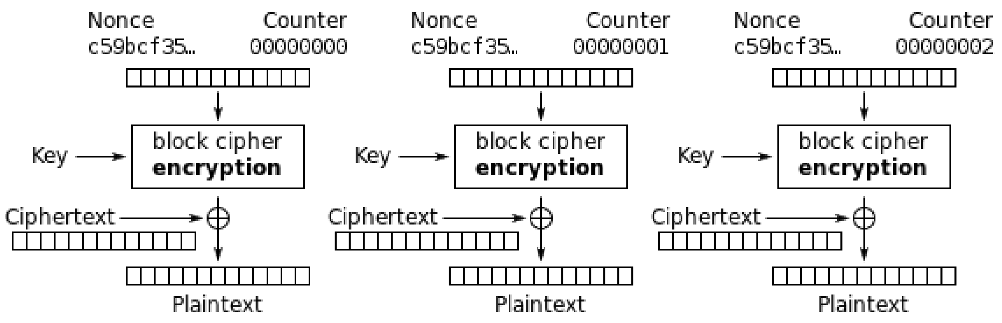
Output Feedback (OFB) mode encryption



Output Feedback (OFB) mode decryption



Counter (CTR) mode encryption



Counter (CTR) mode decryption

Reference: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

2. Command Line Arguments

Program can be executed by command line arguments. Order of arguments are:
“FileCipher -e -i input.txt -o -output.txt algorithm mode key.txt”

In command line arguments, order of arguments are fixed. Also FileCipher, -e, -i, -o arguments are fixed. Names of input file, output file and key file can be anything. Algorithm can be DES or 3DES (Despite prebuilt Cipher class takes DESede as argument for name of 3DES, no need to write DESede instead of 3DES since program handles it automatically) and mode can be CBC, CFB, CTR, and OFB.

After execution;
- Program prints important byte arrays to console. (Importance of it will be discussed in the next parts of report)
- Program updates log file and creates a run.log file if it doesn’t exist.
- Program updates output file and creates that file if it doesn’t exist.
- Program doesn’t change content of input files.

3. Input Files

Program uses one input file for message and another one for keys. Input file can contain multiple lines, Turkish characters, uppercase and lowercase characters, special characters etc. DES and 3DES takes input size as multiple of 8 bytes but our program supports each of input sizes.

Key file should contain 3 keys (they can any length from 1 to ∞ since program handles size differences automatically) but they should be separated with "-". Also spaces between keys are not important, program automatically handles it too. Most important thing is, (as stated in PDF) all input and output files should / will be located at the same directory.

Note: Eclipse works with packages. In src folder, there is a folder named FileCipherV2 which contains source codes. Since package names in source codes is FileCipherV2, we added that folder in order to prevent risk of getting compilation errors.

4. Weaknesses of Program

Four encryption modes that we implemented and two encryption algorithms that imported from Crypto API works without problems. But due to some character set problems, in some cases there can be imperfections in decrypted text files.

4.1.1 - Additional Spaces at the End

If input size is not multiple of 8, program will fill inputs with automatically until it will be multiple of 8. Therefore this padding process, after decryption process you can see spaces at the end.

4.1.2 - Handling Additional Spaces at the End

This can be prevented with .strip(...) or .replaceall(...) methods of Java (and we did it while testing) but if we do it unfortunately program crashes in cases that key size is quite small (such as 1-2 bytes). Since we are not sure about you will try the program with small sized keys, we left the program as it is.

4.2.1 - Imperfections In Decrpyted File Content
After decryption of encrypted content, there can be imperfections (for example weird characters instead of a character) can be written in the decrypted file.

4.2.2 - Main Cause of Imperfections In Decrypted File Content
As you stated in Piazza, a quality program should have support for different characters and Turkish characters. In order to provide that functionality and also prevent some exceptions about special characters, we used 2 different character sets. We tried so many character sets with different combinations and only 1 combination (which stated at below) was worked (produced meaningful output).

Reading Input Files -> UTF_8
Writing Encrypted File -> ISO_8859_1
Reading Encrypted File -> ISO_8859_1
Writing Decrpyted File -> UTF_8

Actually, we can give %99 guarantee about correctness of our implementations of encryption modes and encryption algorithms. This situation is caused by difference of byte arrays after writing and reading encrypted file.

Clue-1: At each of important stages, if we print current byte arrays we can see what causes this. For example:

```
Original Bytes: [[108, 111, 114, 101, 109, 105, 112]]
Encrypted Bytes: [-103, -25, -14, -115, 2, 90, -106]
Converted Bytes (From File): [[-103, -25, -14, -115, 2, -45, -106]]
Decrypted Bytes: [108, 111, 114, 101, 109, 48, 112, 0]
```

Here, original bytes is byte array just after converting input string into byte array. Encrypted bytes is byte array just after encryption. Then program writes encrypted file. Converted bytes is byte array just after reading string from encrypted file and converting them into byte array. Due to some technical problems about encoding of character set by computer some data changes happens here. (Which are colored red at top). Still, program can encode it mostly correctly but at these red bits some weird characters will be seen.

We included exactly same print functions (printing arrays at those 4 important steps) in the source code. Therefore after each execution, if you will see any corruption in output file, then you can compare these byte arrays in order to see cause of this issue.

Clue-2: If we decrypt encrypted content (yes converted strings or byte arrays no matter) without writing and reading it into a file, then no issue occurs.

4.3 - Our Suggestion

We tested program in quite many cases, mostly it works. But we suggest that, program should be tested in different situations. :)