CS 5158/6058 Data Security and Privacy, Spring 2018
Project 2: Symmetric-Key Encryption

Instructor: Dr. Boyang Wang

Due Date: 03/08/2018 (Thursday), 11:59pm.
Format: Please submit a zip file of your code in Blackboard.
Total Points: 10 points
Note: This is an individual project.

1 Project Description

In this project, you will need to implement a symmetric-key encryption scheme using AES. More specifically,

- For the key generation function, you program should be able to output a secret key sk of AES, and write this secret key to a file. By default, the length of your key should be 256 bits and the encryption mode is Cipher Block Chaining (CBC).
- For the encryption function, given a data file f, your program should be able to read a secret key sk, encrypt data file with secret key k, and encrypt this file with this key using AES.

$$c \leftarrow \mathsf{AES.Enc}_k(f)$$

- For the decryption function, given an encrypted data file c and a secret key sk, your program should be able to decrypt this encrypted data file, and write the result of this decryption to a file.

$$f \leftarrow \mathsf{AES.Dec}_k(c)$$

2 Basic Requirements

Programming Language: You can use either C/C++, Python or Java. If you choose to use C/C++, CMake is recommended (but not required). You can choose any IDE you like, the code you submit should be able to compile and run in Linux or Windows. Please provide a clear description about how to compile and run your code in the Readme file.

Crypto Libraries: You can leverage third-party libraries, such as `openssl` (C/C++), `BouncyCastle` (Java), etc., to implement AES encryption. You do not need to build the functions of AES by yourself.

Program Directory: Please name your project folder as `aes_m123456`, where `aes` is the name of this project and `m123456` is your UCID. The recommended directories of your program should be organized as follows:

```
./aes_m123456/src
./aes_m123456/build
./aes_m123456/data
./aes_m123456/Readme.txt
```

Normally, folder `src` should include all the source files and your own header files, e.g., `.cpp` and `.h` files. All the object files and executable files, e.g., `.o` files, should be under folder `build`. Folder `data` has all the given files and data, and also includes all the files and results generated by the program. In `Readme.txt` file, you should write a description of your code, show which language and version you use, and illustrate how to compile, run and use your code.

3 Project Details

For AES encryption, you should choose AES-256-CBC. A (default) plaintext file is stored in "`../data/plaintext.txt`". And the default plaintext in this file is

```
Welcome to data security and privacy.
```

1. Key Generation Function:

(a) Generate a secret key using AES, and write this secret key *sk* to file "`../data/key.txt`". The key needs to be written in hexadecimal (i.e., based 16).

2. Encryption Function:

(a) Read a secret key *sk* from file "`../data/key.txt`", and read a plaintext file *f* from file "`../data/plaintext.txt`";
(b) Generate a random initialization vector *iv*, encrypt plaintext file *f* with secret key *k* and initialization vector *iv*, and write its ciphertext *c* to file "`../data/ciphertext.txt`" in hexadecimal. This random initialization vector is stored in file "`../data/iv.txt`" in hexadecimal.
(c) Your encryption function should be able to encrypt any plaintext file using this encryption function.

3. Decryption Function:

(a) Read a secret key *sk* from file "`../data/key.txt`", an initialization vector *iv* from file "`../data/iv.txt`", and a ciphertext file *c* from file "`../data/ciphertext.txt`";
(b) Output a plaintext file *f* by decrypting *c* with secret key *k*, and write this result of this decryption to file "`../data/result.txt`". The decryption result should be written in (human-readable) string, not in hexadecimal, and it should be the same as the original plaintext in "`../data/plaintext.txt`" if you decrypt correctly.

4. Different Encryption Modes:

(a) Given plaintext file *f* from file "`../data/plaintext.txt`", encrypt this plaintext file in two modes, one in Electronic Codebook (ECB) mode and one in CBC mode, with a same secret key. Repeat your encryption 2 times with each mode, compare your result and explain the difference between the ciphertexts of those two methods. You should have a function in your project to output ciphertexts with a same key but using those two different modes.
(b) You need to submit a one-page report (in pdf) analyzing this comparison between ECB and CBC. In addition, you also need to test the average encryption time and decryption time of CBC mode on the default file, and include those results in your report. In your report, you should describe details of your implementation, such as OS, programming language, crypto libraries, encryption parameters, etc. You can use tables, figures or screenshots to help you present your results and comparison in your report. Please put this report under your project folder and submit it together with your code.

4 Evaluation

Your project will be evaluated in three aspects.

1. **Correctness of Functions (75%):** Your program should be able to correctly run all the functions described in this project. If for some reason, your code cannot be compiled but the logic of your code is correct, you will still get partial credits.

2. **Comments and Descriptions (15%):** Write comments and explain each function in your code, such as inputs, outputs, etc. You may need some of the functions in other projects. Detailed comments on each function can save your time in other projects. In addition, please clearly explain how to compile and run your code in `Readme.txt`. For graduate students, the scores of your report will be included in this aspect.
3. **Coding Style (10%):** A good coding style is always important, especially for large projects. Please keep each function simple, try to avoid long functions, and create multiple `.h` and `.cpp` files if needed. For example, it is not a good idea to put everything in the main function.

5 Examples

This section provides some examples, which can help you understand the functions of your project. If your code can provide the same functionalities, you can customize the number of arguments and the order of arguments, as long as you describe it clearly in the Readme file.

Example 1: The following command calls the key generation function

```
aes  keygen  ../data/key.txt
```

where `aes` is the name of your executable file, `keygen` is the argument for key generation function, `../data/key.txt` is the secret key file. Note that, depending on where your executable file is and your OS, the paths of your input and output files might be different. Security parameter $\lambda = 256$ is default and pre-defined in your program.

Example 2: The following command calls the encryption function

```
aes  enc  ../data/key.txt ../data/plaintext.txt
../data/ciphertext.txt
```

where `../data/plaintext.txt` is the plaintext file, and the ciphertext c will be written to file `../data/ciphertext.txt`. By default, an initialization vector iv is stored in file `../data/iv.txt`.

Example 3: The following command calls the decryption function

```
aes  dec  ../data/key.txt ../data/ciphertext.txt
../data/result.txt
```

where the decryption of a ciphertext will be written to file `../data/result.txt`.