CS 5153/6053 Network Security, Spring 2024
Project 1: Advanced Encryption Standard

Instructor: Dr. Boyang Wang

Due Date: 2/19/2024 (Monday), 11:59pm.
Format: Please submit a zip file of your code in Canvas.
Total Points: 12 points
Note: This is an individual project. Please start your project early and do not wait until the last day to work on your project.

1 Project Description

In this project, you will need to implement the **1st round of encryption** in Advanced Encryption Standard. More specifically,

- Given a message with 128 bits, two subkeys subkey0 and subkey1, your program should be able to perform one AddKey before Round 1 and the corresponding operations (SubBytes, ShiftRows, MixColumns, and AddKey) in Round 1, and output the result of the encryption after Round 1. (In this project, we assume S-box which is used in SubBytes is already given and you can find a copy of it from Lecture 5. We also assume an encryption key has 128 bits in this project.)
- (**Additional Task for CS6053**) Given a 128-bit encryption key, your program should be able to generate the first two subkeys, subkey0 and subkey1 according to subkey schedule algorithm.

If you can implement one round of AES, you should be able to easily extend your program to multiple rounds and have a complete version of AES encryption. In this project, **you only need to focus on the 1st round**, and there is no need to extend it into multiple rounds.

2 Basic Requirements

Programming Language: You can use either C/C++, or Python. You can choose any IDE you like, the code you submit should be able to compile and run in Linux or Windows.

Program Directory: Please name your project folder in the form of `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/report.pdf
```

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 `report.pdf` file, you should describe which OS you use, show which language and version you use, and illustrate how to compile, run and use your code. In addition, you also need to include screenshots for the outputs of each function in your report (please see details in the next section).

3 Project Details

Assume a plaintext file is stored in "`../data/plaintext.txt`". A message is in this file is

```
Two One Nine Two
```

Note that this message has exactly 128 bits (see the example we mentioned in Lecture 5). And two subkeys (in hexadecimal) are given in "`../data/subkey_example.txt`"

```
5468617473206d79204b756e67204675
e232fcf191129188b159e4e6d679a293
```

The subkey on the first line is subkey0 and the subkey on the second line is subkey1. Each has 128 bits (or 16 bytes in hex).

1. First Round of AES Encryption:

(a) Read a message from file "`../data/plaintext.txt`", change each character into ASCII. After that, you should have 128 bits and obtain an initial state (check the example we mentioned in Lecture 5).
(b) Read the two subkeys from file "`../data/subkey_example.txt`", calculate one AddKey before Round 1 with subkey0, compute operations in Round 1 (includes SubBytes, ShiftRows, MixColumns and one AddKey with subkey1). The matrix for MixColumns can be found in Lecture 5, and how to perform the corresponding multiplication over bytes can be found in Lecture 10.
(c) Print your result after the 1st round of AES encryption in terminal and write the result to a file "`../data/result.txt`". The result needs to be printed and written in hexadecimal.
(d) Take a screenshot for the output of your function in step (c) and include it in your report.
(e) You can use the example we mentioned in Lecture 5 to examine the correctness of your function. But your function should be generic and output results for other messages.

Note: for the polynomial additions and multiplications over bytes in $GF(2^8)$, you should implement them by yourself based on what we discussed in lecture 9 and 10.

2. (Additional Task for CS6053) Subkey Schedule:

(a) Read the first subkey from file "`../data/subkey_example.txt`", generate the next subkey using subkey schedule algorithm in AES. Print the next subkey in terminal and write the result to a file "`../data/result_subkey.txt`". The result needs to be printed and written in hexadecimal.
(b) Take a screenshot for the output of your function in step (a) and include it in your report.
(c) You can use the example we mentioned in Lecture 6 to examine the correctness of your function. But your function should be generic and output results for other subkeys.

4 Evaluation

Your project will be evaluated in two aspects.

1. **Correctness of Functions (80%):** 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. **Report (20%):** Please clearly explain how to compile and run your code in `report.pdf`. Please include clear description regarding how you complete the project and clear screenshots for each function.