SENG 360 Assignment 2 – Cryptography

This assignment is based on a similar assignment by Bill Young at UT.

To be worked on in teams of 2-3 students
(Use the course forum or class time to self-organize.)

Objective

To understand the Advanced Encryption Standard (AES) cipher in-depth by implementing it in Java.

Background

The Advanced Encryption Standard (AES) is a very important commercial block cipher algorithm, designed to replace the earlier DES.

You learned about the internals of AES in the online lectures during the week of Sept. 25.

AES uses repeat cycles or ``rounds." There are 10, 12, or 14 rounds for keys of 128, 192, and 256 bits, respectively. The input text is represented as a 4 x 4 array of bytes. The key is represented as a 4 x n array of bytes, where n depends on the key size.
Each round of the algorithm consists of four steps:

1. **subBytes:** for each byte in the array, use its value as an index into a fixed 256-element lookup table, and replace its value in the state by the byte value stored at that location in the table. You can find the table and the inverse table on the web.
2. **shiftRows:** Let Ri denote the ith row in state. Shift R0 in the state left 0 bytes (i.e., no change); shift R1 left 1 byte; shift R2 left 2 bytes; shift R3 left 3 bytes. These are circular shifts. They do not affect the individual byte values themselves.
3. **mixColumns:** for each column of the state, replace the column by its value multiplied by a fixed 4 x 4 matrix of integers (in a particular Galois Field). This is the most complex step. The posted video lectures explain that step in detail. You can also find details at many websites, e.g., Wikipedia. Note that the inverse operation multiplies by a different matrix.
4. **addRoundkey:** XOR the state with a 128-bit round key derived from the original key K by a recursive process.

For encryption, the final round is slightly different from the others and there is an additional addRoundKey. For decryption, the initial round is slightly different and there is an additional addRoundKey.

Assignment Details

Your assignment is to implement AES-256 in Java. That means that you will use 128-bit input, 256-bit key, 14 rounds. There is a tremendous amount of information on the web about the AES algorithm. You can find it by searching for "Rijndael" or "Advanced Encryption Standard" on the web. The official AES standard is here: AES standard. Any questions about the algorithm can be resolved by looking there. Here is some additional info on Key Schedule and additional info on mixColumns.

***Note**: you will likely be able to find complete Java implementations of AES-256 in the Web. Do not just copy them. This would be considered plagiarism and a serious academic integrity issue. You are expected to write your own program. Plus, everyone in your group is expected to contribute and understand the program, and be able to answer questions on it.*

Requirements

1. The primary file name for this assignment is `AES.java`. The main method must exist in `AES.java`. Students can add more files as they wish but all files must be submitted.

2. The assignment must be compiled by `javac *.java`.

3. The assignment must be executed by the following command:

   ```
   java AES option keyFile inputFile
   ```

   where `keyFile` is a key file, `inputFile` (no extension) names a file containing lines of plaintext, and option is "e" or "d" for encryption and decryption, respectively. `keyFile` contains a single line of 64 hex characters, which represents a 256-bit key. The `inputFile` should have 32 hex characters per line. (*For simplicity, we will not worry about padding and just assume that the input fits exactly into multiple lines of 32 hex characters.*)

   - If option "e" is given, `inputFile` is encrypted with the key from `keyFile` and the program outputs an encrypted file named by adding extension ".enc" to the `inputFile`.
   - If option "d" is given, `inputFile` is decrypted with a key from `keyFile` and the program outputs an encrypted file named by adding an extension ".dec" to the `inputFile`. Please be sure that your encrypted file follows the same input format and your decrypted file must match your original plain text (modulo dealing with any malformed or short lines).

     As an example, if you run: `java AES e key plaintext`

     The output should be in file `plaintext.enc`.

     If you then run: `java AES d key plaintext.enc`

     The output would be in `plaintext.enc.dec`, and should match the original `plaintext` input file.

4. The assignment must be compiled by `javac *.java`.

5. Operating mode: use Electronic Code Book (ECB) mode. That is, encode each block separately, without chaining blocks together.

Further explanation

Just to be clear, your input file might start with the following lines:

```
0A935D11496532BC1004865ABDCA4295
00112233445566778899AABBCCDDEEFF
```

....

You'll read in a line, converting from Hex to binary for storage into your state array. Apply the AES algorithm to encrypt the string as stored, and write out the ciphertext in Hex notation to the output file. When decrypting you'll reverse the process. If any line contains non-Hex characters or has more or less than 32 Hex characters (128-bits), through an exception and terminate. Your program should be able to deal with uppercase, lowercase or mixed input.

Deliverable & Details

Submit a zipped archive of your program together with a README file that lists the names of all members of your group (along with any other information needed to compile and run your program) to Courspaces by October 12th.