

## COM 5335 ASSIGNMENT #2 (updated 3/17)

DUE BY 11:59PM 4/2/2022 (Sat)

10% penalty will be applied to late submissions received between 0:00 and 11:59PM 4/3, 20% penalty will be applied to late submissions received between 0:00 AM and 11:59PM 4/4, and 30% penalty will be applied to late submissions received between 0:00 AM and 11:59PM 4/5. No submission will be accepted after 0:00 AM 4/6/2022.

### Objective

Implement the Advanced Encryption Standard (AES).

### Description

First, implement GF(256) as follows. We use 8 bits to represent a polynomial of degree at most 7 as explained in the class. We can also use 8 bits to represent a monic polynomial of degree 8. For example,  $m(x)=x^8+x^4+x^3+x+1$  can be represented as 1b. Represent GF(256) as  $F_2[x]/m(x)$ . Implement the AES encryption function using the following field operations functions (shown in C/C++, but you may use other programming languages, too. Check with the TAs):

```
uint8_t GF256_add(uint8_t a, uint8_t b, uint8_t mx);
// returns a + b. mx is the irreducible polynomial

uint8_t GF256_mult_x(uint8_t a, uint8_t mx);
// Multiplied by x. mx is the irreducible polynomial

uint8_t GF256_mult(uint8_t a, uint8_t b, uint8_t mx);
// General multiplication: mx is the irreducible polynomial

uint8_t GF256_inv(uint8_t *a, uint8_t mx);
// Returns the multiplicative inverse of a. mx is the irreducible polynomial
```

Implement AES as follows.

```
void AES_Encrypt(uint8_t* Plaintext, uint8_t* Ciphertext, uint8_t* Key);
```

Write a main function that calls `AES_Encrypt()`. **Show all intermediate states of each round as well as the final result (round 10) in hexadecimal representation.** Note that you MUST NOT use table look-ups for SubBytes. Compute the multiplicative inverse by calling `GF256_inv()`. In order to make your program I/O more readable, it is required to organize all data in hex numbers.

### Grading

Your program MUST BE compatible with GNU compilers. If you are using other compilers, please make sure your final program is compatible. **You will get no points if your program is not compilable using the abovementioned compilers.** If your program is compilable but the result is not completely correct, you'll still get partial credits. Your program should be well-commented, well-structured, and easy to understand. You may lose up to 30% of points if you fail to do so.

### Submission

Put all your source codes in a folder containing main functions, function implementations, class definitions, or compilation instructions, if any. Compress them as a single zip file. DO NOT submit executable files. Name your zip file as your student ID number (i.e. 100012345.zip). Submit your source code on eLearn.

### Sample I/O (input shown in bold face)

Plaintext: **6e33547730346b5f3565437572315479**

Key: **78686f74ab206d65203e756e6720d67c**

Round 1:

3b 52 75 11 6e 96 94 74 08 7b f8 6a 1b 95 e5 7e

Round 2:

ea bf c5 73 54 c7 a3 ab 1d 22 d8 d4 4b 03 12 46

Round 3:

bf a9 28 d2 21 7a dc 9a 9f 01 0c 47 8c fd aa 7f

Round 4:

b1 ed 3b b2 0d 76 09 29 29 7e 19 40 df 09 ab bc

Round 5:

62 b7 fd 9b 8f 9f a7 f7 80 f2 bc 11 1a 12 11 b6

Round 6:

53 c5 43 4d d0 f4 ed 56 48 1a 13 b6 1e 6b ea e9

Round 7:

f7 2e 1b 8a d6 66 5c d0 94 c1 73 e5 6b 80 ca 08

Round 8:

cf b8 3e 71 2f d0 71 5c 97 ff 0d 46 1b c8 fb 00

Round 9:

70 b6 37 ce bd 62 ab fc 91 eb be 68 6c 2e 5e e7

Round 10:

68 46 4b fd 42 ec f5 65 f8 1b 48 7b 4d 99 9e f8