ECE 404 Homework #4

Due: **THURSDAY** 2/10/2022 at 5:59PM

In this homework, you will get a deeper understanding of finite fields of the form $GF(2^n)$ and the Advanced Encryption Standard (AES).

Theory Problems

1. Determine the following in GF(11), please show your work:
   (a) $(8x^4 + 7x^3 + 2x^2 + 10) + (9x^4 + 6x^3 + x + 7)$
   (b) $(3x^3 + 4x + 3) \times (2x^3 + x^2 + 9x + 7)$
   (c) $\frac{8x^5+6x^4+2x^3+x^2+6x}{2x^3+4x^2+3}$
2. For the finite field $GF(2^3)$, calculate the following for the modulus polynomial $x^3 + x + 1$, please show your work:
   (a) $(x^2 + x + 1) \times (x + 1)$
   (b) $(x + 1) - (x^2 + x + 1)$
   (c) $\frac{x^2+x+1}{x^2+1}$

Programming Problem

Write a script in Python or Perl to implement the AES algorithm with a 256-bit key size.

Instructions:

The following points may aid you in your implementation and are worth noting:

1. Each round of AES involves four steps:
   (a) Single-byte based substitution
   (b) Row-wise permutation
   (c) Column-wise mixing
   (d) Addition of the round key
2. **The order in which these four steps are executed is different for encryption and decryption. The last round for encryption does not involve the 'Mix columns' step. Similarly, the last round for decryption does not involve the 'Inverse mix columns' step.**

3. As you know, AES has variable key-length, and the number of rounds of processing depend upon the key-length. The lecture assumes a 128-bit key length and all subsequent explanation is based upon that assumption. But the key provided to you is 256 bits long, hence, there will be a slight variation in how you generate the *key schedule*. The following explanation will be helpful in that regard:

   (a) For the key expansion algorithm, note that irrespective of the key-length, each round still uses only 4 words from the *key schedule*. Just as we organised the 128-bit key in 4 words for key-expansion, we organise the 256-bit key in 8 words.

   (b) Each step of the key-expansion algorithm takes us from 8 words in one round to 8 words in the next round. Hence, 8 such steps will give us a 64-word key schedule. The implementation of the $g(\cdot)$ function remains the same. The logic of obtaining the 8 words from the j^{th} step of key expansion to the $(j+1)^{th}$ step also remains the same.

   (c) Note that since the key is 256-bits long, there will be 14 rounds of processing in the AES, plus the initial processing. Because each round of processing uses only 4 words from the *key schedule*, you will require only a 60-word *key schedule*. However, the previous step generates a 64-word schedule, so you can ignore the last 4 words in the schedule.

Program Requirements

Similar to your DES implementation in Homework 2, your program should have the following call syntax:

```
python AES.py -e message.txt key.txt encrypted.txt
python AES.py -d encrypted.txt key.txt decrypted.txt
```

Explanation of the syntax:

- **For Encryption** (denoted by `-e`):
  - `AES.py` reads the input plaintext (in this case `message.txt`) and the key (in this case `key.txt`) using the specified file names in the second and third command-line arguments.
  - The encrypted output is then written in **hexstring** form to the file specified by the fourth command-line argument (in this case `encrypted.txt`).
- **For Decryption** (denoted by `-d`):
  - Read in the ciphertext from the second command-line argument (in this case `encrypted.txt`) in **hexstring** format, decrypt it using the specified key, and save the decrypted output to the file specified by the fourth command-line argument (in this case `decrypted.txt`).

Your implementation should not include any external modules (e.g. numpy) other than BitVector and the standard modules already included in the Python Standard Library.

Your decrypted output is allowed to have zero-padding at the very end, and your encryption should pad zeros accordingly just like in your DES implementation.

Useful Notes

- This should go without saying, but **start on this homework early**. This homework is a bit more involved than the previous homeworks.
- On the Brightspace webpage, we have provided the AES results for the plaintext from HW02 after each of the four steps in the **first round of processing for the first block**. The key used is to get these results is: *applesbananaspeachesstrawberries*
- Make sure that when reading the ciphertext during decryption, you need to convert it to a hex-string format and not use the ASCII characters themselves (i.e. do not simply use $BitVector(filename = \ 'encrypted.txt')$ without any more processing). You may want to use $BitVector(hexstring = \ ...)$ instead.
- Keep in mind that the block size is still 128 bits despite the key size being 256 bits.
- You can verify your encryption using the following website (make sure to change the key size and select Hex output) :
  https://www.devglan.com/online-tools/aes-encryption-decryption

  Due to different padding methods for blocks that are not a multiple of the block size, the final block of encryption generated from this website will not match your final encrypted block depending on whether you are padding zeros from the right. However, the remaining blocks should match.

Submission Instructions

- Make sure that the program requirements and submission instructions are followed. **Failure to follow these instructions may result in loss of points!**.
- For this homework you will be submitting 2 files electronically. Your submission must include:
  - A PDF `hw04.pdf` containing:
    - Your answers to the theory problems. You are allowed to include scans or photos of handwritten work in the PDF, but your work must be clearly legible.
  - The file `AES.py/pl` containing your code for the programming problem.
  - If you decide to create any additional code files (e.g. containing helper functions) for your implementation, you may also include up to two additional `.py/.pl` files in your submission for a total of up to 4 files.
- In your program file, include a header as described on the ECE 404 Website.
- If using Python, please denote the Python version in your code with a shebang line (e.g. #!/usr/bin/env python3)
- Please comment your code.

Electronic Turn-in

`turnin -c ece404 -p hw04 hw04.pdf AES.pl` (if using Perl)
`turnin -c ece404 -p hw04 hw04.pdf AES.py` (if using Python)