ECE 404 Homework #4

Due: Tuesday 2/18/2020 at 4:29PM

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

IMPORTANT: For this homework, you will have both a **physical** (i.e. paper hard-copy) and **electronic** submission. The physical submission should be handed in at the front of the classroom on the due date. See the Submission Notes section for details.

Theory Problems

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

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

1. Each round of AES involves the following:
   (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. 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 the 128-bit key length and all subsequent explanation is based upon that assumption. The key provided to you is 256-bits long and hence there will be a slight variation in how you generate the *keyschedule*. 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 *keyschedule*. 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 $g()$ 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-bit 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 *keyschedule*, you will require only a 60-word *keyschedule*. However, the previous step generates a 64-word schedule,so you will have to 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 , `AES.py` reads the input plaintext from `message.txt` and the key from `key.txt`, then writes the encrypted output in **hexstring** form to `encrypted.txt`
For decryption (denoted by `-d`), read in the ciphertext `encrypted.txt` in **hexstring** format and save the decrypted output to `decrypted.txt`.
Make sure when reading the encrypted text during decryption that you convert it to the hex values they represent and not the ASCII characters themselves (i.e. do not simply do $BitVector(filename = 'encrypted.txt')$ without any more processing). You may want to use $BitVector(hexstring = ...)$ instead.

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 ECE 404 Homework webpage, we have provided a sample plaintext. We have also provided the results 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: $thepurdueuniversityboilermakers$!
- You can check to see if your encryption works properly 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 if you pad zeros from the right. However, the remaining blocks should match.

Submission Instructions

- Make sure to follow program requirements and submission instructions. **Failure to follow these instructions may result in loss of points!**.
- Your paper submission must include your answers to the theory problems and a printout of your code.
- For the electronic submission, turn in only the file containing your program (e.g. AES.py)
- If using Python, please denote the Python version in your code with either a shebang line (e.g. #!/usr/bin/env python3)
- Please comment your code.

Electronic Turn-in

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