ECE 404 HW 4

Nimal Padmanabhan

February 14, 2023

1 Theory Problems

1. Determine the following in GF(11), please show your work:

   (a) $(9x^5 + 4x^4 + 8x^3 + 2x^2 + 3x + 4) + (6x^5 + 2x^4 + 9x^3 + 7x^2 + 5x + 7)$

   $15x^5 + 6x^4 + 17x^3 + 9x^2 + 8x + 11$
   Mod terms that are above GF(11) by 11
   $\boldsymbol{4x^5 + 6x^4 + 6x^3 + 9x^2 + 8x}$

   (b) $(8x^3 + 6x^2 + 8x + 1) \times (3x^3 + 9x^2 + 7x + 5)$

   $24x^6 + 90x^5 + 134x^4 + 157x^3 + 95x^2 + 47x + 5$
   Mod terms that are above GF(11) by 11

   $\boldsymbol{2x^6 + 2x^5 + 2x^4 + 3x^3 + 7x^2 + 3x + 5}$

   (c) $\frac{3x^3-5x^2+10x-3}{3x+1}$
   Rewrite dividend in terms of GF(11):
   $\frac{3x^3+6x^5+10x+8}{3x+1}$
   $\frac{3x^3}{3x} = 3 * MI(3) = 3 * 4 = 12\%11 = 1$
   First term in quotient: x^2
   First subtraction: $3x^3 + 6x^2 + 10x + 8 - (3x^3 + x^2) = 6 + AI(5) + 10x = 5x^2 + 10x$
   $\frac{5x^2}{3x} = 5 * MI(3) = 5 * 4 = 20\%11 = 9$
   Second term in quotient: $9x$
   Second subtraction: $5x^2 + 10x - (5x^2 + 9x) = x - 3$
   Third term: 4
   Third subtraction: $x - 3 - (x + 4) = -7\%11 = 4$
   Answer: $\boldsymbol{x^2 + 9x + 4 + \frac{4}{3x+1}}$

2. For the finite field GF(2^3), calculate the following for the modulus polynomial $x^3 + x + 1$, please show your work:

   (a) $(x^2 + x + 1) \times (x^2 + x)$
   $x^4 + x^3 + x^2 + x^3 + x^2 + x^2 + x$
   $x^4 + x$
   Perform polynomial long division using the modulus polynomial and find the remainder
   $\frac{x^4+x}{x^3+x+1}$
   $\frac{x^4}{x^3} = 1 * MI(1) = 1$
   First term in quotient: x
   First subtraction: $x^4 + x - (x^4 + x^2 + x) = -x^2 = x^2$
   Final answer: $\boldsymbol{x^2}$

   (b) $(x^2) - (x^2 + x + 1)$
   $x^2 - x^2 - x - 1$
   $-x - 1$
   $\boldsymbol{x + 1}$

   (c) $\frac{x^2+x+1}{x^2+1}$

   $\frac{x^2+x+1}{x^2+1}$
   $\frac{x^2}{x^2} = 1 * MI(1) = 1$
   First term in quotient: 1
   First subtraction: $x^2 + x + 1 - (x^2 + 0x + 1) = x$
   Final answer: $\boldsymbol{1+\frac{x}{x^2+1}}$

2 Programming Problem

For this programming problem, we were tasked to implement the 256-bit version of the Advanced Encryption Standard (AES) algorithm for both encryption and decryption. AES operates on a block cipher size of 128 bits and has three modes of operation for the key size: 128, 192, and 256 bits. The only thing that changes with the key size in AES is how the key schedule is generated from the key. For encryption and decryption in the 256-bit key mode, it consists of 14 rounds of processing. For the first 13 rounds in encryption, each round of processing includes 1 single-byte based substitution step, 1 row shift permutation step, 1 column-wise mixing step, and the addition of the round key. On the last round of the encryption, the mixing of the columns does not occur. For the decryption side, the order is slightly different and the round keys will be reversed to decrypt the encrypted message file. As a result, the first 13 rounds in decryption include inverse shift rows, inverse substitute bytes, add round key, and inverse mix columns. Just like encryption, the last round of decryption does not include the inverse mix columns step.

Now we will go through an overview of each step in both encryption and decryption. Even though the order of steps are not the same for both parts, the algorithm is more or less the same process for each step. For substitution of bytes, we use the 4x4 state array that we have XORed with the first four words from the round key list and populated the array from the bits from the file to populate the global list variable `subTableBytes` to perform the bit scrambling done in the `genTables()` function. For decryption, it is the same procedure except we now use the `invSubTableBytes` global list variable. For the `shift_rows()` function, we shift the rows circularly to the left by the current row number minus one for encryption using the modulus operator. For decryption, it is the same idea but now we do circularly shift to the right by the current row number minus one. When we reach the `col_mix()` function, we need to perform modular multiplication taking the current element of the two-dimensional state array as a `BitVector` object and multiplying them with the appropriate constants using the `gf_modular_multiply()` function from the `BitVector` library. For encryption, the terms will multiplied by `0x2` and `0x3` while for decryption, the terms will be the hex values of `0x0e`, `0x0b`, `0x0d`, and `0x09`. When these constants will be multiplied with certain numbers will be dependent on the row number. After doing the modular muliplication, their products will be XORed together to perform the column mixing of the state array for each row. When adding the round keys, we take the current state array as a `BitVector` object and XOR it with the round keys from the current round plus one to account for the XORing step we did before the substitution of bytes step. After going through all rounds in both encryption and decryption, we either produce an encrypted text file in hex characters or a decrypted text file of the message in ASCII characters.