Scott Christensen

09/05/2016

C.S. 465

H.W. #2: Finite Field Arithmetic

1. MixColumns():

First, create a temporary holder for your result which is the same size as your original column.

Then we will perform matrix multiplication using out original column and our predefined size appropriate given matrix, then save these results in the temporary holder.

To do this matrix multiplication, we use the XOR bit shift multiplication technique we saw in class.

mixColumns(state){

for each col in state{

resultCol = new byte(4);

resultCol = fMult(0x02, Col(0), XOR

fMult(0x03, Col(1), XOR

fMult(0x01, Col(2), XOR

fMult(0x01, Col(3), XOR));

= fMult(0x01, Col(0), XOR

fMult(0x02, Col(1), XOR

fMult(0x03, Col(2), XOR

fMult(0x01, Col(3), XOR));

= fMult(0x01, Col(0), XOR

fMult(0x01, Col(1), XOR

fMult(0x02, Col(2), XOR

fMult(0x03, Col(3), XOR));

= fMult(0x03, Col(0), XOR

fMult(0x01, Col(1), XOR

fMult(0x01, Col(2), XOR

fMult(0x02, Col(3), XOR));

}

return state;

}

2. FMulti():

We take in two bytes. We’ll make a temp copy of our second byte. For any given two bits, you’ll loop through the following process 8 times (because you’re iterating over 8 bits): Left shift the first byte by 1. If you and it with 0X80 you return the XOR with 0X1B. If you don’t just do a bit shift.

fMult(){

fMulti(byteOne, byteTwo);

|  |
| --- |
| byteTemp = byteTwo; |
| byteTotal = 0x00; |
| for(int i = 0; i < 8; i++){ |
| if(((byteOne . (i))& 0x01) == 0x01) |
| byteTotal ^= byteTemp;  if(byteTemp &0x80 == 0x80)  byteTemp = byteTemp << 1 ^ 0x1B;  else  byteTemp = byteTemp << 1; |
|  |
| } |
| return byteTotal; |

}