**Christopher LaJon Morgan**

**Brother Seamons**

**CS465 Homework #2**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* This function will take each column of the state matrix and multiply

\* it by a fixed matrix.

\*

\* The fixed matrix is:

\* [ 02, 03, 01, 01]

\* [ 01, 02, 03, 01]

\* [ 01, 01, 02, 03]

\* [ 03, 01, 01, 02]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **AESCypher::mixColumns**()

{

**for** (**int** c = 0; c < 4; c++)

{

**unsigned** **char** ans[4];

**for** (**int** c = 0; c < 4; c++)

{

ans[0] = ffMultiply(0x02, state[0][c]) ^

ffMultiply(0x03, state[1][c]) ^

state[2][c]^

state[3][c];

ans[1] = state[0][c] ^

ffMultiply(0x02, state[1][c]) ^

ffMultiply(0x03, state[2][c]) ^

state[3][c];

ans[2] = state[0][c] ^

state[1][c] ^

ffMultiply(0x02, state[2][c]) ^

ffMultiply(0x03, state[3][c]);

ans[3] = ffMultiply(0x03, state[0][c]) ^

state[1][c] ^

state[2][c] ^

ffMultiply(0x02, state[3][c]);

state[0][c] = ans[0];

state[1][c] = ans[1];

state[2][c] = ans[2];

state[3][c] = ans[3];

}

**return**;

}

//======Field Arithmetic Helpers

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Calls xtime on num till the multiple of two in numTimes is reached.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**char** **AESCypher::xtimes**(**char** num, **char** numTimes)

{

**char** times = 0x01;

**while** (times != numTimes)

{

num = xtime(num);

times = times << 1;

}

**return** num;

}

//======Field Arithmetic

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Performs xor on one and two

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**char** **AESCypher::ffAdd**(**char** one, **char** two)

{

**return** one ^ two;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Adds x to fField

\* ie: left shift clear high bit

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**char** **AESCypher::xtime**(**char** fField)

{

**return** (fField << 1) & 0x7f;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Performs fixed field multiplication on bytes one and two.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**char** **AESCypher::ffMultiply**(**char** one, **char** two)

{

**bool** set = **false**;

**char** ans = 0x00;

**for** (**int** i = 0; i < 8; i++,two = two >> 1)

{

**if** (two & 0x01)

{

**if** (!set)

{

set = **true**;

ans = xtimes(one, BitPositions[i]);

}

**else**

ans = ffAdd(ans, xtimes(one, BitPositions[i]));

}

}

**return** ans;

}

**const** **char** BitPositions[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};