B5 - Mathematics

B-MAT-500

my_PGP Bootstrap

Let's code AES

my_PGP Bootstrap

binary name: AES
language: everything working on "the dump"
compilation: when necessary, via Makefile, including re, clean and fclean rules

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

The difficult part of the PGP project is the AES cryptosystem.
The algorithm is divided into several steps in order to mix the data.
Some of the steps are pretty easy, others are more mixed up.

1. KeyExpansion is the step where you build subkeys from the original key.

2. Then there is 4 steps per round:

   - SubBytes : substitution of every element through a lookup table (you can find the table here.
   - ShiftRows : each row of the matrix is shifted of 0, 1, 2 or 3 positions.
   - MixColumns : mixing the columns with matrix multiplication.
   - AddRoundKey : XOR with the subkey(from the KeyExpansion step)

As there is several ways to implement each of the parts, but it is not easy to know if you get the correct output, you will find some reference outputs to help you get through the different steps.

KeyExpansion

First step of AES cryptosystem is to construct the subkeys (11 round keys for AES 128).
key = [0x10, 0x50, 0xa9, 0x25, 0x15, 0xd6, 0x55, 0x55, 0xd4, 0x50, 0xeb, 0x45, 0x68, 0x21, 0xe9, 0x81]
key 0 = [16, 80, 169, 37, 21, 214, 85, 85, 212, 80, 235, 69, 104, 33, 233, 129]
key 1 = [236, 78, 165, 96 , 249, 152, 240, 53, 45, 200, 27, 112, 69, 233, 242, 241]
key 5 = [231, 140, 34, 169, 229, 145, 39, 33, 205, 39, 201, 56, 174, 68, 209, 10]
key 9 = [111, 175, 89, 174, 160, 104, 17, 181, 55, 224, 141, 101, 162, 103, 191, 32]
key 10 = [220, 167, 238, 148, 124, 207, 255, 33, 75, 47, 114, 68, 233, 72, 205, 100]

SubBytes and ShiftRows

Here is the output of the first steps, with the previous key, and the following message
message = [0x14,0x15,0x16,0x17,0x10,0x11,0x12,0x13,0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1F,0x20]
xor (key[0], message) = [4, 69, 191, 50, 5, 199, 71, 70, 204, 73, 241, 94, 116, 60, 246, 161]
xor then Subbytes then shift = [242, 198, 161, 50, 107, 59, 66, 35, 75, 235, 8, 90, 146, 110, 160, 88]

MixColumns and Round

Here is the output of the first round
output = [209, 225, 158, 110, 3, 96, 65, 183, 207, 12, 69, 250, 48, 4, 189, 34]

Full AES

At the end of the rounds but the final one
output = [8, 57, 71, 243, 158, 131, 41, 188, 68, 232, 158, 26, 173, 103, 167, 213]
All the rounds + final = [236, 75, 229, 151, 119, 84, 163, 44, 80, 170, 210, 33, 124, 90, 104, 198]

Decipher

Now that you managed to code a AES cipher for a block, you can code the decipher part, and see if you can retrieve your original message.