CMPE 494 Assignment 1

Group Members:
Çisel Zümbül
Fikri Cem Yılmaz
Salih Bedirhan Eker

Github Repo Link:
https://github.com/salih997/CMPE-494-AES

How does AES work?

AES is by now the most important symmetric algorithm in the world. AES is a block cipher, each block is 128 bits. Key is 128/192/256 bits. AES works in an iterative way (10/12/14 iteration). Each iteration (round) consists of 4 layers. Byte-substitution, shift rows, mix columns and add round keys. Last round does not include mix columns layer.

Byte-substitution layer provides confusion, shift rows and mix column layers provide diffusion.

What happens inside these layers?
At first you need to create a round key from the initial key.
1-Byte-sub:
This layer is actually just a substitution operation.We take input as a two digit hex number. The first digit is one axis and the second digit is another axis. Then we have a precalculated table called S-box (16x16 0 to f). We just substitute the value according to this table.

2-Shift rows:
These layers shift rows of the 4x4 matrix. It shifts zero left first row, one shift left second row, two shifts left third row and three shifts left the last row.

3-Mix columns:
This layer basically is a matrix multiplication. Each column is multiplied with a specific matrix and replaced.

4-Add round keys:
XOR the processed matrix until now and corresponding round key. At this point, new round keys are calculated according to the ‘AES Key Schedule’.

Note that when all rounds are finished, the algorithm just encrypts one block. Continue until all input data is encrypted.

Decryption
For decryption you need to reverse the steps basically. The order is as follows:
Add round key, Inverse MixColumns, Shift rows, Inverse Sub-byte.
Differences:
In Inverse mix column step matrix will be different.
In shift row step, shifts will be right shift.
In sub-byte step use reverse S-box table.

Architecture of our program

In our program, we have 3 options for key length as 128, 192 and 256 bits and 2 options for the mode of the encryption, ECB and CBC. Our program also supports the decryption of the ciphertext with the options above.

The AES algorithm uses 10, 12 and 14 rounds according to the given key length. Each round has 4 principal steps. First, substitution of bytes; second, shifting rows; third, mixing columns and finally, adding the round key. Adding the round key part also has its own key schedule and each round key is created according to the keys created before. In our program we implemented these steps as separate functions.

At the beginning of our program, we encrypted/decrypted our text in blocks of 128 bits. We had four 32 bits words. We placed them in a 2 dimensional integer array where each entry represented a byte. Since we needed to have our bytes placed in vertical fashion to our 4x4 matrix we implemented a transpose function and used it many times. We applied the steps mentioned above on this base matrix, round times.

Implementing the key schedule, we created a key words arraylist that kept track of each round's ‘keywords’ that are byte arrays of size 4. We implemented the key schedule algorithm, using this key words list. Each round we created our new keywords using the keywords created before. According to the algorithm, to create the keyword of our key, we generally XORed the keyword that is the one before and the keyword “(key length)/32” before in the key words list. In decryption, we calculated all key words at the beginning of the program, filled up our key words list and used them four by four, starting from the end of the list.

In AES algorithm main adding functionality is XOR, so for adding keys we’ve simply used built in bitwise XOR function.

For mixing columns, we’ve implemented matrix multiplication, where it basically does bitwise shifting for multiplication and XOR function for adding.

For sub bytes, we’ve prepared a lookup table and sub the bytes according to that table.

How many MB/sec can we encrypt/decrypt?

We can encrypt and decrypt
62164 lines = 62164 * (16 bytes) = 994624 bytes = 0.9485 MB/sec if we don't print intermediate steps to the screen.

However, it changes according to input size.