Introduction to Cryptology and Coding
Lab 2: AES, finite field

SID-LAKHDAR Riyane

June 5, 2017

Contents

1 Introduction **2**

2 Multiplication of a triple by a matrix **2**

3 SubBytes on a triple **2**

4 Key schedule **3**

5 Ciphering **3**

6 Improve the efficiency of the AES **3**

1 Introduction

This document resumes and explains the program that we have implemented to cipher and decipher message using the advanced encryption standard (AES). This program (attached to this document) may be run by simply running the two following commands:

- javac AES.java
- java AES

Then each question of the topic will be printed separately with an explanation.

2 Multiplication of a triple by a matrix

For this task, we have implemented the in the class **"Tools-F-2-power-3"** the function
multiplyMatrixByPoly(int[] matrix, int poly, int addPoly, int matrixXSize) Where each line of the matrix is represented by an entry of the input array, and where the vector (the triple) is simply an integer.
All the computations are done modulo 2. Thus, we don't need to compute any multiplication or addition. The algorithm will simply compute logical and between each line of the matrix and the vector. The corresponding line in the vector will simply 1 or 0 depending on whether the number of "1" in the result is even or odd.

3 SubBytes on a triple

To implement the function subBytes, we first need to be able to inverse an element of $\mathbb{F}_{2^3}$.
To do so, we can notice that the elements of $\mathbb{F}_{2^3}$ form a finite group with respect to multiplication: $a^{p^n-1} = 1$ (where $p = 2$ and $n = 3$). Thus $a^{-1} = a^6$
The inverse of an element of $\mathbb{F}_{2^3}$ is done by the function **inverse** in the class **Tools-F-2-power-3**. This function will simply compute a^6 for en entry a, and return the rest of its division by 11 (a^6 and 11 are polynomials. The modulo function of two polynomials is implemented in the same class).

Then the subByte function of an input a is given by the formula $A * y + x$ where y is the inverse of a in $\mathbb{F}_{2^3}$, and A and c are the two given matrix and vectors (see function **AES.subByte**).

The full S-box for the input in $\mathbb{F}_{2^3}$ is: $5, 2, 6, 1, 0, 0, 3, 2$

4 Key schedule

The key schedule function has been implemented in **AES.keySchedule**. The scheduled key computed by our method is

$$\begin{aligned} \text{Key} &= 011 \mid 100 \mid 100 \mid 001 \\ \text{Key round 0} &= 011 \mid 100 \mid 100 \mid 001 \\ \text{Key round 1} &= 001 \mid 101 \mid 001 \mid 000 \end{aligned} \tag{1}$$

5 Ciphering

We have implemented the different steps of the AES algorithm: ShiftRow, MixColumns and AddRoundKey.
Thanks to this functions (and to them inverse), we have implemented the cipher functions (and the decipher) in the java function class AES.cipher.
The result is:

$$\begin{aligned} \text{Original message} &= 001 \mid 001 \mid 110 \mid 011 \\ \text{Ciphered message} &= 010 \mid 001 \mid 110 \mid 110 \end{aligned} \tag{2}$$

6 Improve the efficiency of the AES

We can notice that each step of the AES algorithm is deterministic. We can also notice (by printing the successive keys and ciphered messages) that many combinations of key and messages are repeated.
Thus to make our algorithm more efficient, we have stored for each possible input message the result of the successive operations: subByte, shiftRow, and mixColumn.
To do so, we have needed for each input message (2^{12} different matrix) one only matrix of the same size. Thus the memory needed is 2^{12} times the size of a message. Which in our implementation represents a total memory of 65536 bytes.

We have on purpose ignored the addRoundKey operation because this operation depends on the key at each round. Hence, it would make the memory needed sky rocket.

The three operations listed bellow had a complexity of $O(n^2)$, where n is the size of the message. Thus by replacing each of them by a simple lookup in a table, the complexity of the algorithm would become constant (assuming that the xor operation needed for the addRoundKey is done in constant time).