# AES OPTIMIZATION: 32-bit platforms

March 20, 2016

## 1 Introduction

We shall discuss some aspects regarding optimization of AES for 32 bit platforms. We assume that the reader is familiar with AES description [1].

## 2 Optimization

The different steps of the round transformation can be combined in a single set of look-up tables for very fast implementation on processors with word lengths 32 or greater.

Let the input of the round transformation be denoted by **a**, and the output of `SubBytes` by **b**:

$$b_{i,j} = S[a_{i,j}], 0 \leq i \leq 4; 0 \leq j \leq N_b \tag{1}$$

Let the output of `ShiftRows` be denoted by **c** and the output of `MixColumns` by **d**:

$$\begin{pmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{pmatrix} = \begin{pmatrix} b_{0,j+C_0} \\ b_{1,j+C_1} \\ b_{2,j+C_2} \\ b_{3,j+C_3} \end{pmatrix}, 0 \leq j < N_b \tag{2}$$

$$\begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{pmatrix}, 0 \leq j < N_b. \tag{3}$$

Equations 1-3 can be combined into:

$$\begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} S[a_{0,j+C_0}] \\ S[a_{1,j+C_1}] \\ S[a_{2,j+C_2}] \\ S[a_{3,j+C_3}] \end{pmatrix}, 0 \leq j < N_b.$$

The matrix multiplication from can be interpreted as a linear combination of four column vectors:

$$\begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} = \begin{pmatrix} 02 \\ 01 \\ 01 \\ 03 \end{pmatrix} S[a_{0,j+C_0}] \oplus \begin{pmatrix} 03 \\ 02 \\ 01 \\ 01 \end{pmatrix} S[a_{1,j+C_1}] \oplus \begin{pmatrix} 01 \\ 03 \\ 02 \\ 01 \end{pmatrix} S[a_{2,j+C_2}] \oplus \begin{pmatrix} 01 \\ 01 \\ 03 \\ 02 \end{pmatrix} S[a_{3,j+C_3}]$$

We define now the four $T$-tables: $T_0, T_1, T_2$ and $T_3$:

$$T_0[a] = \begin{pmatrix} 02 \cdot S[a] \\ 01 \cdot S[a] \\ 01 \cdot S[a] \\ 03 \cdot S[a] \end{pmatrix}$$

$$T_2[a] = \begin{pmatrix} 03 \cdot S[a] \\ 02 \cdot S[a] \\ 01 \cdot S[a] \\ 01 \cdot S[a] \end{pmatrix}$$

$$T_2[a] = \begin{pmatrix} 01 \cdot S[a] \\ 03 \cdot S[a] \\ 02 \cdot S[a] \\ 01 \cdot S[a] \end{pmatrix}$$

$$T_3[a] = \begin{pmatrix} 01 \cdot S[a] \\ 01 \cdot S[a] \\ 03 \cdot S[a] \\ 02 \cdot S[a] \end{pmatrix}$$

These tables have each 256 4-byte word entries and require $4kB$ of storage space. Using these tables, we can write:

$$\begin{pmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{pmatrix} = T_0[a_{0,j+C_0}] \oplus T_1[a_{1,j+C_1}] \oplus T_2[a_{2,j+C_2}] \oplus T_3[a_{3,j+C_3}], 0 \leq j < N_b$$

Taking into account that `AddRoundKey` can be implemented with an additional 32-bit XOR operation per column, we get a look-up table implementation with $4kB$ of tables that takes only four look-ups and four XOR operations per round.

Furthermore, the entries $T_0[a], T_2[a], T_2[a]$ and $T_4[a]$ are rotated versions of one another, for all values $a$. Consequently, at the cost of three additional rotations per round per column, the look-up table implementation can be realized with only one table, i.e. with a total table size of $1kB$. The size of the encryption routine (relevant in applets) can be kept small by including a program to generates the tables instead of the tables themselves.

# References

[1] Federal Information Processing Standard 197.