Exercise 6.6

Task

This concerns a trick that is very useful in implementing AES. Let S be the AES S-box function, i.e., it inputs and outputs a single byte. Let MC be the mix-column function, i.e., it takes as input a column consisting of 4 bytes and outputs such a column. Note that MC is linear, i.e., we have for any two columns $C1$, $C2$ that $MC(C1 \oplus C2) = MC(C1) \oplus MC(C2)$. Define a function T_0 as follows: it takes as input a byte b, and the output $T_0(b)$ is a 4-byte column computed as follows: you first form a column $C(b)$ by placing $S(b)$ in the top byte and all-0 bytes in the lower three positions. Then set $T_0(b) = MC(C(b))$. We also define functions T_1,T_2,T_3. They are similar to T_0, except that when we form $C(b)$, we place $S(b)$ in the second, respectively third, respectively fourth entry from the top, and put in 0's elsewhere. Now consider the state of an AES encryption at the start of some round. Name the bytes in this state a_{ij} and let R be the state after we have done `SubBytes`, `ShiftRow` and `MixColumn`. So R is a 4 by 4 matrix of bytes. Show that the first column of R is $T_0(a_{00}) \oplus T_1(a_{11}) \oplus T_2(a_{22}) \oplus T_3(a_{33})$ Give similar expressions for the other 3 columns of R. Sketch how this result can be used to implement AES based only on table look- up and XOR, instead of explicitly computing the operations. How much memory would you need for this?

Solution

We have to show that applying `SubBytes` -> `ShiftRows` -> MixColumns‘ will give us $T_0(a_{00}) \oplus T_1(a_{11}) \oplus T_2(a_{22}) \oplus T_3(a_{33})$ in the first column.

Instead of using b_{ij} we'll use $S(a_{ij})$ for the calculations below, but here's what the `SubBytes` function looks like applied on the matrix:

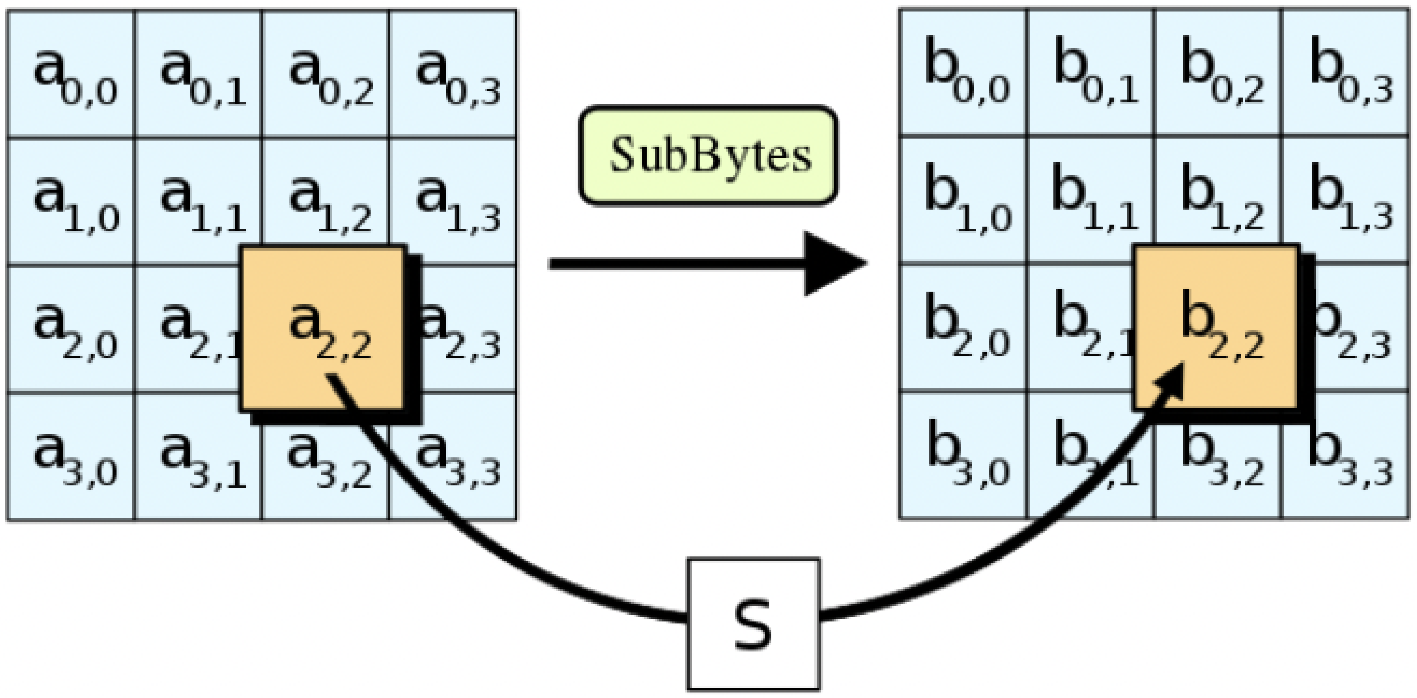

Figure 6.3 The AES SubBytes operation

And for the last step `MixColumns` we apply, where `Col` is each column of the matrix we get after applying `ShiftRows`.

$$M \cdot Col = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \cdot \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\xrightarrow{\text{SubBytes}} \begin{pmatrix} S(a_{00}) & S(a_{01}) & S(a_{02}) & S(a_{03}) \\ S(a_{10}) & S(a_{11}) & S(a_{12}) & S(a_{13}) \\ S(a_{20}) & S(a_{21}) & S(a_{22}) & S(a_{23}) \\ S(a_{30}) & S(a_{31}) & S(a_{32}) & S(a_{33}) \end{pmatrix}$$

$$\xrightarrow{\text{ShiftRow}} \begin{pmatrix} S(a_{00}) & S(a_{01}) & S(a_{02}) & S(a_{03}) \\ S(a_{11}) & S(a_{12}) & S(a_{13}) & S(a_{10}) \\ S(a_{22}) & S(a_{23}) & S(a_{20}) & S(a_{21}) \\ S(a_{33}) & S(a_{30}) & S(a_{31}) & S(a_{32}) \end{pmatrix}$$

$$\xrightarrow{\text{MixColumns}} \begin{pmatrix} xS(a_{00}) \oplus (x+1)S(a_{01}) \oplus S(a_{02}) \oplus S(a_{03}) & ... & ... & ... \\ S(a_{00}) \oplus xS(a_{01}) \oplus (x+1)S(a_{02}) \oplus S(a_{03}) & ... & ... & ... \\ S(a_{00}) \oplus S(a_{01}) \oplus xS(a_{02}) \oplus (x+1)S(a_{03}) & ... & ... & ... \\ (x+1)S(a_{00}) \oplus S(a_{01}) \oplus S(a_{02}) \oplus xS(a_{03}) & ... & ... & ... \end{pmatrix} = R$$

Now, let's see what we get for T_i:

$$T_0(b) = MixColumns \begin{pmatrix} S(b) \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} xS(b) \\ S(b) \\ S(b) \\ (x+1)S(b) \end{pmatrix}$$

$$T_1(b) = MixColumns \begin{pmatrix} 0 \\ S(b) \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} (x+1)S(b) \\ xS(b) \\ S(b) \\ S(b) \end{pmatrix}$$

$$T_2(b) = MixColumns \begin{pmatrix} 0 \\ 0 \\ S(b) \\ 0 \end{pmatrix} = \begin{pmatrix} S(b) \\ (x+1)S(b) \\ xS(b) \\ S(b) \end{pmatrix}$$

$$T_3(b) = MixColumns \begin{pmatrix} 0 \\ 0 \\ 0 \\ S(b) \end{pmatrix} = \begin{pmatrix} S(b) \\ S(b) \\ (x+1)S(b) \\ xS(b) \end{pmatrix}$$

If we expand $T_0(a_{00}) \oplus T_1(a_{11}) \oplus T_2(a_{22}) \oplus T_3(a_{33})$ we'll see that it is exactly first column of R.

$$T_0(a_{00}) \oplus T_1(a_{11}) \oplus T_2(a_{22}) \oplus T_3(a_{33}) = \begin{pmatrix} xS(a_{00}) \oplus (x+1)S(a_{01}) \oplus S(a_{02}) \oplus S(a_{03}) \\ S(a_{00}) \oplus xS(a_{01}) \oplus (x+1)S(a_{02}) \oplus S(a_{03}) \\ S(a_{00}) \oplus S(a_{01}) \oplus xS(a_{02}) \oplus (x+1)S(a_{03}) \\ (x+1)S(a_{00}) \oplus S(a_{01}) \oplus S(a_{02}) \oplus xS(a_{03}) \end{pmatrix}$$

Using similar calculations we can show that

$$R_{i0} = T_0(a_{00}) \oplus T_1(a_{11}) \oplus T_2(a_{22}) \oplus T_3(a_{33})$$
$$R_{i1} = T_0(a_{01}) \oplus T_1(a_{12}) \oplus T_2(a_{23}) \oplus T_3(a_{30})$$
$$R_{i2} = T_0(a_{02}) \oplus T_1(a_{13}) \oplus T_2(a_{20}) \oplus T_3(a_{31})$$
$$R_{i3} = T_0(a_{03}) \oplus T_1(a_{10}) \oplus T_2(a_{21}) \oplus T_3(a_{32})$$

AES algorithm is as follows

1. Setup
   - Given plaintext X
   - Set State = X
   - set State = $AddRoundkey(State, K_0) = State \oplus K_0$.
2. For $i = 1 \; to \; N-1$, do:
   - $State = SubBytes(State)$
   - $State = ShiftRows(State)$
   - $State = MixColumns(State)$
   - $State = AddRoundkey(State, K_i) = State \oplus K_i$
3. In the final round, do
   - $State = SubBytes(State)$
   - $State = ShiftRows(State)$
   - $State = AddRoundkey(State, K_N) = State \oplus K_N$

Step 1 uses only XOR operations.

Step 2 we just showed how we can implement using the XOR operation and a substitution table for the $S(b)$ function. This works because $S(b)$ is a function that works on 8-bit inputs, so it's a 1:1 deterministic mapping, hence it can be replaced with a finite substitution table.

Step 3 `SubBytes` is a table lookup. `ShiftRows` is bit-shifting, which can be implemented using a table-lookup or probably even XOR, but I see no reason to do so as bit-shifting operations are the fastest operations in modern CPUs. Therefore, step 3 can also be implemented only using XOR and table lookups.

The space we need to allocate for substitution tables is mainly for the T_i functions. Each of the 4 T_i takes in 1b input and outputs 4b output. The possible inputs are 2^8 and the T_i are all deterministic functions. Therefore, the total memory is $M(T_i) = O(4 \times 2^8 \times 4) = O(2^{12}) bytes$