Advanced Encryption Standard (AES)

PROJECT REPORT

The procedural step by step working of the project code has been explained:

Step 1. Original Data: 128 bits of data is taken in the form of 2D array.

Step 2. Starting Key: 128 bits of key is taken at random.

Step 3. Add round key: Simply put, the starting key is added(XOR) to the original data.

Step 4. A combination of different procedures is performed (10 times for this program ie. 10 rounds).

a. *Byte substitution*: Read an element from data, separate its first 4 bits and last 4 bits, and substitute the original data with data in s-Box in such a way that first 4 bits corresponds to row number of s-Box and last 4 bits corresponds to column number.

b. *Row Shifting*: Row shifting is done in such a fashion that first row remains unchanged, in second row each element is given 1 byte circular shift to left, in third row, 2 byte circular shift to left and in fourth row, 3 byte circular shift to left.

c. *Column Mixing*: The data is multiplied with a constant matrix in Galois Field. The constant matrix is:

2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Multiplication can be done quite easily with the use of the two tables in (HEX), mentioned in the program as "E_Table" and "L_Table". Multiplication is explained below with an example:

Consider the first column of constant matrix as [2 , 3 , 1 , 1] and then consider a sample input as [0xD4, 0xBF, 0x5D, 0x30]. Keep in mind that the multiplication is done using the tables and the addition is simple bit-wise XOR.

```
[2  3  1  1]  x  [D4]

                 [BF]

                 [5D]

                 [30]

Output 0       =       (D4  *  02)  XOR   (BF  *  03)  XOR  (5D * 1)  XOR  (30 * 1)

               =       E (L(D4) + L(02)) XOR E(L(BF) + L(03)) XOR  5D  XOR  30
```

= E (41 + 19) XOR E(9D + 01) XOR 5D XOR 30

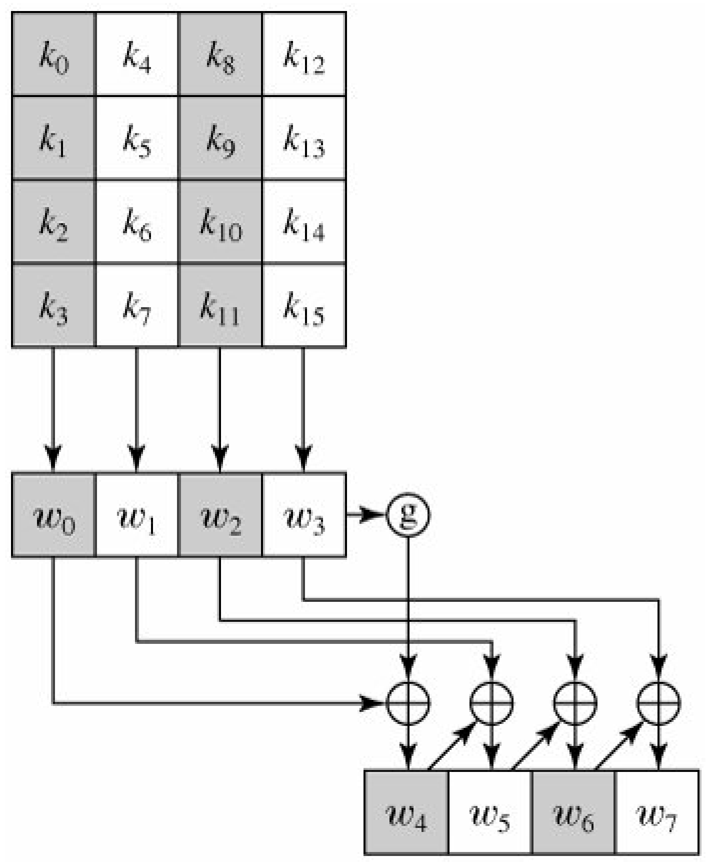
= E (5A) XOR E(9E) XOR 5D XOR 30

= B3 XOR DA XOR 5D XOR 30

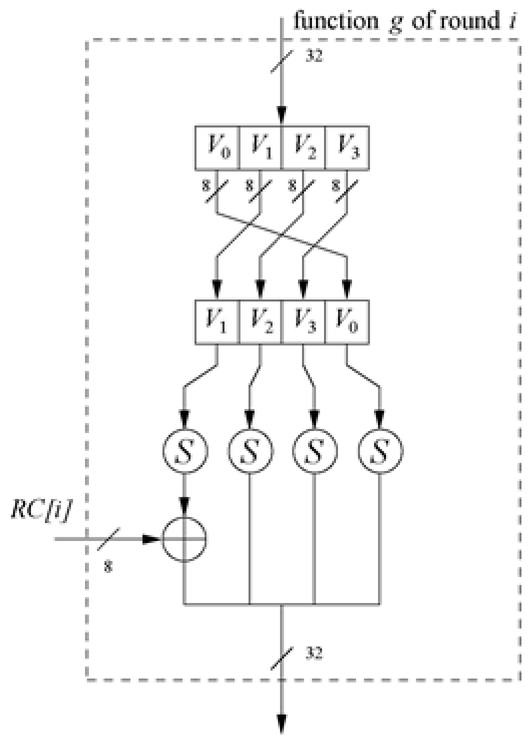
= 04

And so on for the rest of outputs. First the value is substituted by L_Table and then we take the arithmetic sum of the values. If this sum goes beyond 0xFF, we simply subtract 0xFF from the result, and then using the remaining value, we substitute the value using E_Table. In the end, all the values are XOR'ed and we get the final multiplied value in Galois Field.

d. *Round key generation*: Round keys are generated the following way:



The round key for round 1 will be w[4] – w[7] and is generated from the previous round key. It is generated using simple XOR of each words by the next word, and the function “g” in the small box on top can be explained as:



The 32 bit word of the key is just swapped in the way it is shown in the figure, and then each byte is substituted from the s-Box and the first byte is XOR'ed with the ith element of RC array ("i" is the number of round and RC is the array of constants which are calculated in Galois Field).

After the program runs for 10 rounds of above 4 functions, we will get the encrypted output data.