Available online at www.sciencedirect.com

SciVerse ScienceDirect

Physics Procedia 25 (2012) 842 – 847

Physics Procedia

2012 International Conference on Solid State Devices and Materials Science

Multiple Lookup Table-Based AES Encryption Algorithm Implementation

Jin Gong ,Wenyi Liu, Huixin Zhang

Key Laboratory of Instrumentation Science &Dynamic Measurement (North University of China), Ministry of Education,Science and Technology on Electronic Test &Measurement Laboratory
North University of China
Taiyuan, China

Abstract

Anew AES (Advanced Encryption Standard) encryption algorithm implementation was proposed in this paper. It is based on five lookup tables, which are generated from S-box(the substitution table in AES). The obvious advantages are reducing the code-size, improving the implementation efficiency, and helping new learners to understand the AES encryption algorithm and GF(28) multiplication which are necessary to correctly implement AES[1]. This method can be applied on processors with word length 32 or above, FPGA and others. And correspondingly we can implement it by VHDL, Verilog, VB and other languages.

*Keywords:*AES implementation;multiple lookup tables； VHDL

1. Introduction

The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits. This paper only describes the method how to encrypt with multiple lookup tables(without caring key expansion), usingthe 128-bit plaintext and 128-bit key. There aremainly three parts to describe my implementation method. The first part is to present the difference between the usual implementation and mine, the second part will present how to generate the lookup tables which my implementation used, and at last there will have a conclusion.

2. The Difference about Encryption

There involve 4 kinds of 128-bit data transformation in AES encryption algorithm in officialAES

 doi:10.1016/j.phpro.2012.03.166

specification.They are calledSubBytes, ShiftRows, MixColumns, andAddRoundKey. The cipher is presented in the pseudo codein fig.1. And its flow chart is in fig.2 correspondingly.

The multiple lookup table-based AES encryption includes only one kind of transformation, and that is AddRoundKey, presented in fig.3. The simplification results from the five Lookup tables (A0, A1, A2, A3, and A4) used in this method. The Pseudo Code for the multiplelookup table-basedAES encryption algorithm implementation is in fig.4，which also shows where the five tables used.

Compare the difference between the officialimplementation and mine, the simple style is obvious. The next chapter will describe how the lookup tables generate specificly.

```
Cipher (byte plaintext [16], byte ciphertext [16],
word w [44])
Begin
    byte state [16]
    state= plaintext
    AddRoundKey (state, w [0, 3])

    for round = 1 step 1 to 9
        SubBytes(state)
        ShiftRows(state)
        MixColumns (state)
        AddRoundKey (state, w [4*round,
                              4*(Round+1)-1])
    end for
    SubBytes (state)
    ShiftRows (state)
    AddRoundKey (state, w [40, 43])
    Ciphertext = state
end
```

Figure1. Pseudo Code for the official AES encryption [2]

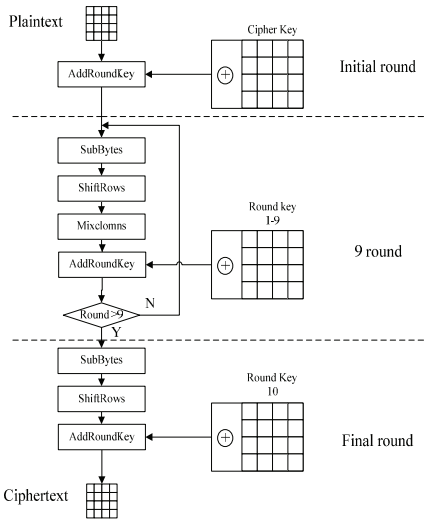

Figure2.Flow chart of the official AES encryption

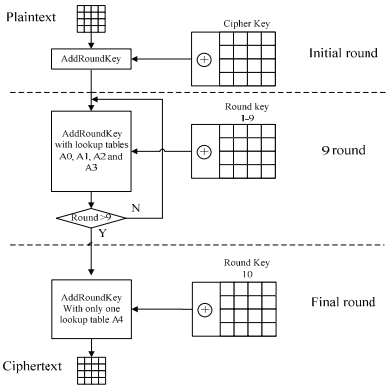

Figure3.Flow chart of the multiple lookup table-based AES encryption algorithm implementation

```
Cipher (byte plaintext [16], byte ciphertext [16],word
        w [44])
Begin
    wordT0, T1, T2, T3
    T0 = W [0] XOR plaintext [0, 3]
    T1 = W [1] XOR plaintext [4, 7]
    T2 = W [2] XOR plaintext [8, 11]
    T3 = W [3] XOR plaintext [12, 15]
    K = 4
    for round = 1 step 1 to 9
        S [0, 3] = T0
        S [4, 7] = T1
        S [8, 11] = T2
        S [12, 15] = T3
        T0=A0(S(0))XORA1(S(5))XOR A2(S(10))XOR A3(S(15)) XORW(k + 0)
        T1=A0(S(4))XORA1(S(9))XORA2(S(14)) XOR A3(S(3)) XOR W(k + 1)
        T2=A0(S(8))XORA1(S(13))XOR A2(S(2)) XOR A3(S(7)) XOR W(k + 2)
        T3=A0(S(12))XORA1(S(1))XORA2(S(6)) XOR A3(S(11)) XOR W(k + 3)
        K = K + 4
    end for
    S[0, 3] = T0
    S[4, 7] = T1
    S[8, 11] = T2
    S[12, 15]= T3
   ciphertext[0, 3]= A4(S(0)) XOR A4(S(5)) XOR
          A4(S(10)) XOR A4(S(15))XORw(k + 0)
  ciphertext[4, 7]= A4(S(4)) XOR A4(S(9)) XOR A4(S(14)) XOR A4(S (3)) XOR W(k + 1)
  ciphertext[8,11] = A4(S(8)) XOR A4(S(13)) XOR A4(S(2)) XOR A4(S(7)) XOR W(k + 2)
  ciphertext[12,15] = A4(S(12)) XOR A4(S(1))
          XOR A4(S(6)) XOR A4(S(11)) XOR
          W(k + 3)
End
```

Figure4. Pseudo Code for the multiple lookup table-based AES encryption algorithm implementation

3. Generation of lookup Tables

In fact, all of the five lookup tables are generated from S-box. Then one circle of the main loop of the AES encryption algorithm will be used to describe the generation. Provided the array in fig.5 is the input array of the SubBytes transformation in the main loop in fig.2.

$S_{0.0}$	$S_{0.1}$	$S_{0.2}$	$S_{0.3}$
$S_{1.0}$	$S_{1.1}$	$S_{1.2}$	$S_{1.3}$
$S_{2.0}$	$S_{2.1}$	$S_{2.2}$	$S_{2.3}$
$S_{3.0}$	$S_{3.1}$	$S_{3.2}$	$S_{3.3}$

Figure5.The input array of the SubBytes of the main loop

After the transformation, the output array is shown in fig.6.

$S'_{0.0}$	$S'_{0.1}$	$S'_{0.2}$	$S'_{0.3}$
$S'_{1.0}$	$S'_{1.1}$	$S'_{1.2}$	$S'_{1.3}$
$S'_{2.0}$	$S'_{2.1}$	$S'_{2.2}$	$S'_{2.3}$
$S'_{3.0}$	$S'_{3.1}$	$S'_{3.2}$	$S'_{3.3}$

Figure6.The output array of the SubBytes of the main loop

Then fig.6 is used as the input array of ShiftRows transformation, and the output is in fig.7.

$S'_{0.0}$	$S'_{0.1}$	$S'_{0.2}$	$S'_{0.3}$
$S'_{1.1}$	$S'_{1.2}$	$S'_{1.3}$	$S'_{1.0}$
$S'_{2.2}$	$S'_{2.3}$	$S'_{2.0}$	$S'_{2.1}$
$S'_{3.3}$	$S'_{3.0}$	$S'_{3.1}$	$S'_{3.2}$

Figure7.The output array of the ShiftRows of the main loop

Then it is the turn of MixColumnstransformation which is the core of the generation of the lookup tables. This transformation operates on the input array column-by-column. The four bytes in a column are replaced by the following:

$$S''_{0.c} = (\{02\} \cdot S'_{0.c}) \oplus (\{03\} \cdot S'_{1.c}) \oplus S'_{2.c} \oplus S'_{3.c} \quad (1)$$

$$S''_{1.c} = S'_{0.c} \oplus (\{02\} \cdot S'_{1.c}) \oplus (\{03\} \cdot S'_{2.c}) \oplus S'_{3.c} \quad (2)$$

$$S''_{2.c} = S'_{0.c} \oplus S'_{1.c} \oplus (\{02\} \cdot S'_{2.c}) \oplus (\{03\} \cdot S'_{3.c}) \quad (3)$$

$$S''_{3.c} = (\{03\} \cdot S'_{0.c}) \oplus S'_{1.c} \oplus S'_{2.c} \oplus (\{02\} \cdot S'_{3.c}) \quad (4)$$

(thesubscript c represents the column of the array in fig.7 and fig.8)

Apply (1) - (4) on the array in fig.7, then we will get the array in fig.8 as the output of the MixColumns transformation.

If we combine (1), (2), (3), (4), SubBytes transformation and ShiftRows transformation as thefollowing pseudo code in fig.9, we can get [B0B1B2 B3]which is the same array as that in fig.8. And part of A0, A1, A2, and A3 is shown in fig.10, fig.11, fig.12, fig.13 separately. Before the generation, we shouldexpand S-box into a word table byte by byte as fig.14. And then generate A0, A1, A2, and A3 referencing the pseudo code in fig.9.

$S''_{0.0}$	$S''_{0.1}$	$S''_{0.2}$	$S''_{0.3}$
$S''_{1.0}$	$S''_{1.1}$	$S''_{1.2}$	$S''_{1.3}$
$S''_{2.0}$	$S''_{2.1}$	$S''_{2.2}$	$S''_{2.3}$
$S''_{3.0}$	$S''_{3.1}$	$S''_{3.2}$	$S''_{3.3}$

Figure8.The output array of the MixColumns

$word B_c (c = 0, 1, 2, 3)$
$Provided B_c = s''_{3.c} \& s''_{2.c} \& s''_{1.c} \& s''_{0.c}$
andthen
$B_0 = A0 (S_{0.0}) XOR A1 (S_{1.1}) XOR A2 (S_{2.2}) XOR A3 (S_{3.3})$
$B_1 = A0 (S_{0.1}) XORA1 (S_{1.2}) XORA2 (S_{2.3}) XORA3 (S_{3.0})$
$B_2 = A0 (S_{0.2}) XOR A1 (S_{1.3}) XOR A2 (S_{2.0}) XOR A3 (S_{3.1})$
$B_3 = A0 (S_{0.3}) XORA1 (S_{1.0}) XORA2 (S_{2.1}) XOR A3 (S_{3.2})$

Figure9.Pseudo Code using lookup tables

	0	1	2
0	A56363C6	847C7CF8	997777EE
1	45CACA8F	9D82821F	40C9C989
2	C2B7B775	1CFDFDE1	AE93933D
3	0C040408	52C7C795	65232346

Figure10.Part of A0

	0	1	2
0	6363C6A5	7C7CF884	7777EE99
1	CACA8F45	82821F9D	C9C98940
2	B7B775C2	FDFDE11C	93933DAE
3	0404080C	C7C79552	23234665

Figure11.Part of A1

	0	1	2
0	63C6A563	7CF8847C	77EE9977
1	CA8F45CA	821F9D82	C98940C9
2	B775C2B7	FDE11CFD	933DAE93
3	04080C04	C79552C7	23466523

Figure12.Part of A2

	0	1	2
0	C6A56363	F8847C7C	EE997777
1	8F45CACA	1F9D8282	8940C9C9
2	75C2B7B7	E11CFDFD	3DAE9393
3	080C0404	9552C7C7	46652323

Figure13.Part of A3

	0	1	2
0	63636363	7C7C7C7C	77777777
1	CACACACA	82828282	C9C9C9C9
2	B7B7B7B7	FDFDFDFD	93939393
3	04040404	C7C7C7C7	23232323

Figure14.Part of A4

After the change, the four transformationsarejust simplified into only one, which is AddRoundKey. The description is in fig.4.

Conclusion

The 5 lookup tables can not only combine the different steps of the round transformation so that reduce the code size, but also improve the implementationefficiency. It has a kind of simple style which can help readers study multiplication in GF(28). Besides, the method has a widely using expansion can be implemented in VB, C, VHDL and other languages.

Acknowledgment

This work was sponsored by Project 50675212 supported by National Natural Science Foundation of China.

References

[1] Encrypt It: Keep Your Data Secure with the New AdvancedEncryption Standard. MSDN magazine November 2003

[2] Announcing the ADVANCED ENCRYPTION STANDARD (AES). NIST, 2001:1-26