Application Report
SLAA397A–July 2009–Revised March 2009

AES128 – A C Implementation for Encryption and Decryption

Uli Kretzschmar — *MSP430 Systems*

ABSTRACT

This application report describes the AES algorithm and the use of a suggested C implementation for AES encryption and decryption with MSP430.

Note: This document may be subject to the export control policies of the local government.

Contents

1 Introduction

The Advanced Encryption Standard (AES) was announced by the National Institute of Standards and Technology (NIST) in November 2001. [1] It is the successor of Data Encryption Standard (DES), which cannot be considered as safe any longer, because of its short key with a length of only 56 bits.

To determine which algorithm would follow DES, NIST called for different algorithm proposals in a sort of competition. The best of all suggestions would become the new AES. In the final round of this competition the algorithm Rijndael, named after its Belgian inventors Joan Daemen and Vincent Rijmen, won because of its security, ease of implementation, and low memory requirements.

There are three different versions of AES. All of them have a block length of 128 bits, whereas the key length is allowed to be 128, 192, or 256 bits. In this application report, only a key length of 128 bits is discussed.

1.1 Basic Concept of the Algorithm

The AES algorithm consists of ten rounds of encryption, as can be seen in Figure 1. First the 128-bit key is expanded into eleven so-called round keys, each of them 128 bits in size. Each round includes a transformation using the corresponding cipher key to ensure the security of the encryption.

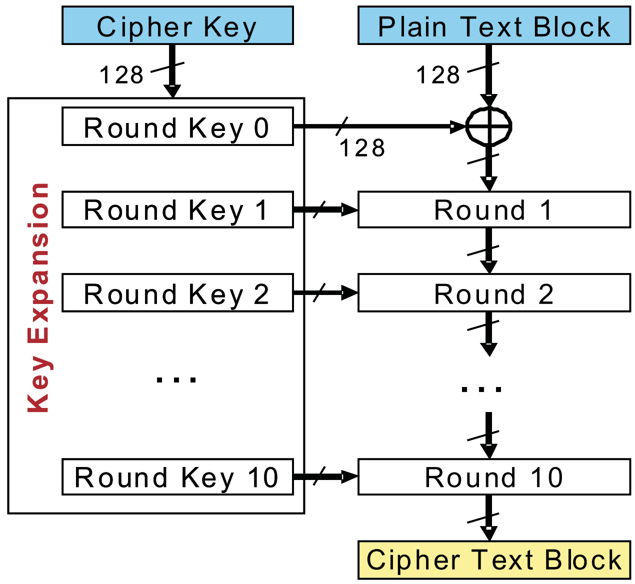

Figure 1. AES Algorithm Structure

After an initial round, during which the first round key is XORed to the plain text (Addroundkey operation), nine equally structured rounds follow. Each round consists of the following operations:

- Substitute bytes
- Shift rows
- Mix columns
- Add round key

The tenth round is similar to rounds one to nine, but the Mix columns step is omitted. In the following sections, these four operations are explained.

1.2 Structure of Key and Input Data

Both the key and the input data (also referred to as the state) are structured in a 4x4 matrix of bytes. Figure 2 shows how the 128-bit key and input data are distributed into the byte matrices.

The State

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

The Key

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

Figure 2. Structure of the Key and the State

1.3 Substitute Bytes (Subbytes Operation)

The Subbytes operation is a nonlinear substitution. This is a major reason for the security of the AES. There are different ways of interpreting the Subbytes operation. In this application report, it is sufficient to consider the Subbytes step as a lookup in a table. With the help of this lookup table, the 16 bytes of the state (the input data) are substituted by the corresponding values found in the table (see Figure 3).

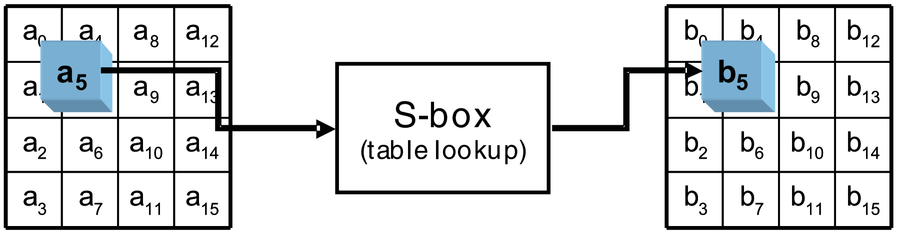

Figure 3. Subbytes Operation

1.4 Shift Rows (Shiftrows Operation)

As implied by its name, the Shiftrows operation processes different rows. A simple rotate with a different rotate width is performed. The second row of the 4x4 byte input data (the state) is shifted one byte position to the left in the matrix, the third row is shifted two byte positions to the left, and the fourth row is shifted three byte positions to the left. The first row is not changed.

Figure 4 illustrates the working of Shiftrows.

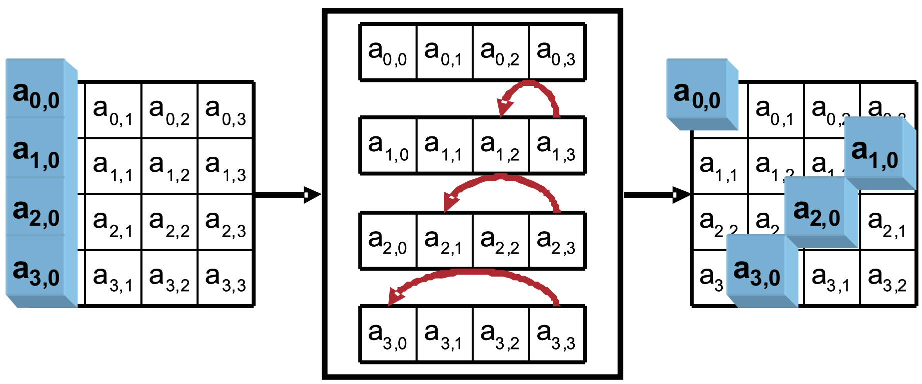

Figure 4. Shiftrows Operation

1.5 Mix Columns (Mixcolumns Operation)

Probably the most complex operation from a software implementation perspective is the Mixcolumns step. The working method of Mixcolumns can be seen in Figure 5.

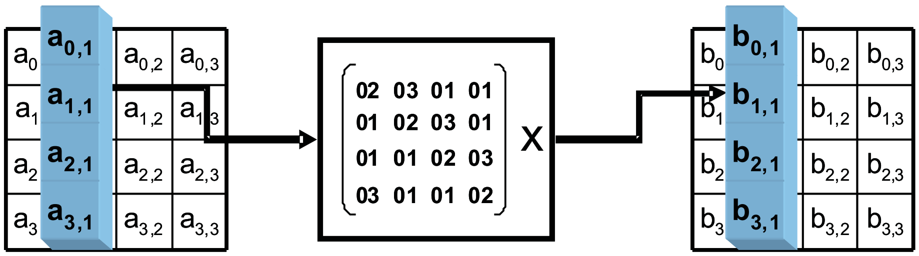

Figure 5. Mixcolumns Operation

Opposed to the Shiftrows operation, which works on rows in the 4x4 state matrix, the Mixcolumns operation processes columns.

In principle, only a matrix multiplication needs to be executed. To make this operation reversible, the usual addition and multiplication are not used. In AES, Galois field operations are used. This paper does not go into the mathematical details, it is only important to know that in a Galois field, an addition corresponds to an XOR and a multiplication to a more complex equivalent.

The fact that there are many instances of 01 in the multiplication matrix of the Mixcolumns operation makes this step easily computable.

1.6 Add Round Key (Addroundkey Operation)

The Addroundkey operation is simple. The corresponding bytes of the input data and the expanded key are XORed (see Figure 6).

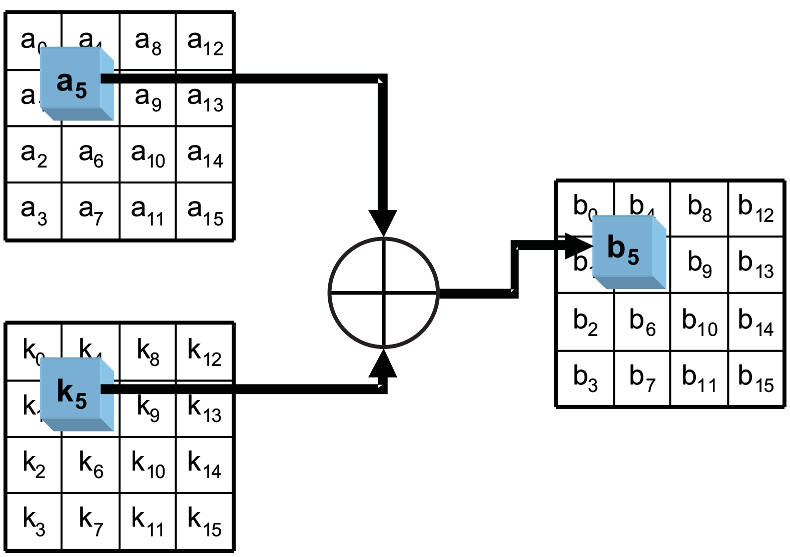

Figure 6. Addroundkey Operation

1.7 Key Expansion (Keyexpansion Operation)

As previously mentioned, Keyexpansion refers to the process in which the 128 bits of the original key are expanded into eleven 128-bit round keys.

To compute round key (n+1) from round key (n) these steps are performed:

1. Compute the new first column of the next round key as shown in Figure 7:

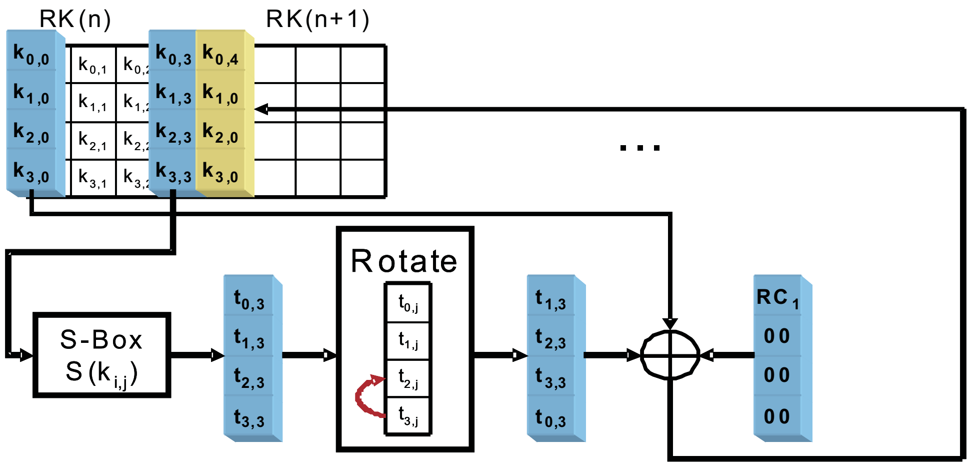

Figure 7. Expanding First Column of Next Round Key

   First all the bytes of the old fourth column have to be substituted using the Subbytes operation. These four bytes are shifted vertically by one byte position and then XORed to the old first column.

   The result of these operations is the new first column.

2. Columns 2 to 4 of the new round key are calculated as shown:
   - [new second column] = [new first column] XOR [old second column]
   - [new third column] = [new second column] XOR [old third column]
   - [new fourth column] = [new third column] XOR [old fourth column]

   Figure 8 illustrates the calculation of columns 2-4 of the new round key.

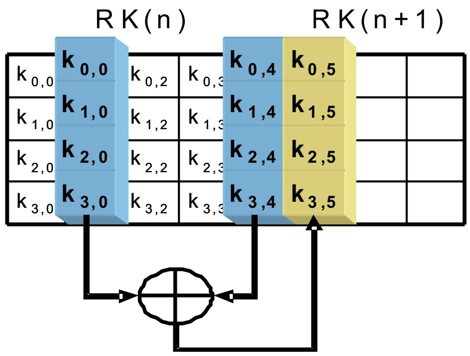

Figure 8. Expanding Other Columns of Next Round Key

2 C Implementation of AES 128

The algorithm was implemented using C. The following sections show how an encryption or decryption can be calculated using the functions provided by this application report.

2.1 Encrypting 128 Bit

The following code example shows how an AES encryption can be performed.

```
#include "msp430x26x.h"
#include "TI_aes.h"

int main( void )
{
  unsigned char state[] = {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
                           0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff};
  unsigned char key[]   = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                           0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f};
  aes_encrypt(state,key);

  return 0;
}
```

This short program defines two arrays of the type unsigned character. Each array is 16 bytes long. The first one contains the plaintext and the other one the key for the AES encryption.

After the function aes_encrypt returns, the encryption result is available in the array state.

2.2 Decrypting 128 Bit

Decryption can be done in a similar way to encryption. First two arrays are defined. When a decryption needs to be performed, one array contains the key and the other one the cipher text.

After the function aes_decrypt returns, the decryption result is available in the array state.

```
#include "msp430x26x.h"
#include "TI_aes.h"

int main( void )
{
  unsigned char state[] = { 0x69, 0xc4, 0xe0, 0xd8, 0x6a, 0x7b, 0x04, 0x30,
                            0xd8, 0xcd, 0xb7, 0x80, 0x70, 0xb4, 0xc5, 0x5a};
  unsigned char key[]   = {0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                           0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f};
  aes_decrypt(state,key);

  return 0;
}
```

3 Including the AES Algorithm In User Code

The AES encryption and decryption software is distributed as source code so that the user can compile this software together with other code. To accomplish the inclusion, some steps should be taken:

- Add TI_aes.c to the project and including the header file TI_aes.h.
- Change the device-specific include file in the c-file of the library.
- Optional step: Change the compiler optimizations to High→Speed to decrease the cycle count of the AES calculation.

4 Performance

In the following table, the performance metrics of the AES-implementation can be found. These values correspond to the optimization setting High→Speed of the C-compiler in IAR.

	Encryption	Decryption
Clock cycles needed	~6600	~8400
Flash usage (bytes)	1839	2423
RAM usage (bytes)	80	80

5 Included Library Files

TI_aes.c – Contains all needed functions for AES en- and decryption

TI_aes.h – Includes the function definitions for AES en- and decryption.

5.1 Function description

void aes_encrypt(unsigned char *state, unsigned char *key)

This function is used for encryption using the AES algorithm. It has the following parameters:

- unsigned char *state

  This is the 16-byte long array containing the plaintext that is to be encrypted. The user should always pass a copy of the plaintext to the encrypt function, because data is overwritten by the internal calculations.
- unsigned char *key

  This 16-byte long array contains the 128-bit key for the AES encryption.

The state array is used for the calculation and contains the result of the encryption, the cipher text, on return of the function.

void aes_decrypt(unsigned char *state, unsigned char *key)

This function is used for decryption using the AES algorithm. It has the following parameters:

- unsigned char *state

  This is the 16-byte long array containing the cipher text that is to be decrypted. The user should always pass a copy of the cipher text to the decrypt function, because data is overwritten by the internal calculations.
- unsigned char *key

  This 16-byte long array contains the 128-bit key for the AES decryption.

The state array is used for the calculation and contains the result of the decryption, the plaintext, on return of the function.

6 References

1. Announcing the Advanced Encryption Standard (FIPS PUB 197)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2009, Texas Instruments Incorporated