Optimization of Memory for AES Rijndeal algorithm Implementation on Embedded system

M. Ravindra Babu and A. R. Reddy

Madanapalle Institute of Technology and Science, Madanapalle, Chittoor, Andhra Pradesh.

Abstract

Advanced Encryption Standard (AES) algorithm has gained popularity as it is deployed in various embedded systems. Realization of AES algorithm on 8051 microcontroller with minimum memory will be useful for deploying it in low cost applications. This paper addresses the memory requirement when the AES algorithm is ported into microcontroller using standard embedded system design tools. Theoretical memory requirement is calculated and is compared with measured values. Algorithm is compiled using KEIL and SDCC compilers targeting into an 8051 microcontroller board. The results are presented for key lengths of 128 bits and 256 bits. The measured results shows higher memory requirement than the theoretical values during the implementation. Also, SDCC yields lowest possible RAM for AES algorithm. These observations further confirmed by porting the compiled program into an embedded board realized with P89V51RD2 controller.

Keywords

Memory Optimization, AES, Embedded System

1. Introduction

AES-Rijndael was developed by Joan Daemen and Vincent Rijmen, Rijndael [4,5] and was selected from five finalists namely; 1) Mars Developed by the IBM team that developed Lucifer, 2) RC6 Developed by the RSA Laboratories, 3) Rijndael Developed by Joan Daemon and Vincent Rijmen, 4) Serpent Developed by Ross Anderson, Eli Biham, and Lars Knudsen, 5) Two fish Developed by Counterpane Systems. The selection was done based upon the parameters such as security, performance, efficiency, flexibility, and implementability by NIST. The selected algorithm, viz., Advanced Encryption Standard (AES), has replaced DES and published as FIPS 197 in November 2001 [5]. It is a symmetric block cipher that can process 128 bits message blocks and 128, 192, and 256 bits key lengths. Both hardware and software implementation of AES-Rijndael are more attractive. Hardware implementation is common in high speed application since architecture provides easy implantation structure. Software implementation [4] is relatively slow and consumes processor time and hence for embedded system application hardware implementation is popularly done.

Since the standardization of the algorithm, AES is implemented in various hardware platforms based upon controllers / processors with 8-bit words to 64-bit words. The AES implementation on the 8-bit controllers is easy with low cost and used for low end applications, whereas large bit processors are expensive and used for high end applications such as MPLS routers, IPv6 routers, etc. This paper explores the reduction of the cost in implementing AES on 8-bit controllers. Two compiler tools are studied for memory allocation required for AES in 8-bit controllers. In order to demonstrate the memory requirements, two different compilers are used to compile the AES algorithm for implementation on controller. One is KEIL, which is popular among the embedded tools and another is Small Device C Compiler (SDCC). The results in each case are obtained and compared. Further, an embedded board with P89V51RD2 microcontroller is used to demonstrate the AES algorithm implementation with lowest possible memory. The results suggest that SDCC is useful for realizing the AES algorithm on low end controllers and processors.

1.1 Basic structure of AES-Rijndael:

A full description of the AES-Rijndael is detailed in the Rijndael proposal [4] and FIPS 197 [5]. Three criteria are taken into account in the design of AES- Rijndael;

1. Resistance against all known attacks.
2. Speed and code compactness on a wide range of platforms.
3. Design simplicity.

In most ciphers, round transformation has the Feistel Structure. But Round transformation of Rijndael does not have the Feistel structure. Instead, the round transformation is composed of three distinct inversable uniform transformations, called layers.

The linear mixing layer: Guarantees high diffusion over multiple rounds. It is achieved using two operations 1) Plain text multiplication with MDS matrix and 2) cyclic shift of bytes of a word.

The non-linear layer: Parallel application of S-boxes that have optimum worst-case nonlinearity properties.

***The key addition layer*:** A simple EXOR operation of the Round Key to the intermediate State.

Rijndael is a very good performer in both hardware and software across a wide range of computing environments regardless of its use in feedback or non-feedback modes

Manuscript received September 5, 2013
Manuscript revised September 20, 2013

[6]. Its key setup time is excellent, and its key agility is good. Rijndael's very low memory requirements make it very well suited for restricted-space environments. Rijndael's operations are among the easiest to defend against power and timing attacks.
Rijndael is designed with some flexibility in terms of block and key sizes, and the algorithm can accommodate alterations in the number of rounds. Finally, Rijndael's internal round structure has instruction-level parallelism and hence implementing on hardware is very easy.
AES is a block cipher developed to address the threatened key size of Data Encryption Standard (DES). It allows the data lengths of 128 bits and three different key lengths, 128, 192, and 256 bits. The number of rounds in each case is decided by the key length [4]. For a plain text of 128 bits number of rounds are as follows,

Key length	Block length in words Nb	Key length in words Nk	Number of rounds Nr
128-bit key	4	4	10
192-bit key	4	6	12
256-bit key	4	8	14

Where N_b is input block length is divided by 32, N_k is the Key length divided by 32 and N_r is the number of rounds. The input bit sequence is first transformed into byte sequence. In next step a two-dimensional array of bytes (called the State) is built. The State array consists of four rows of bytes, each containing 4 bytes. All internal operations (Cipher and Inverse Cipher) of the AES algorithms are performed on the State array. The AES algorithm basically consists of four byte oriented transformation for encryption and inverse transformation for decryption process namely,

a) Byte substitution using substitution box table. (S-box):
b) Shifting rows of the state array using different offsets. (Row transformation)
c) Mixing the data within each column of the state array. (Mixing columns)
d) Adding a round key to the state. (Add round key)

The figure 1 shows AES Rijndael Encryption and Decryption structure, where the input to the encryption and decryption algorithm is 128 bit block. The key provided is expanded into an array of forty - four 32 bit words, w[i]. Four distinct words forming 128 bits serve as a key for each round in both encryption and decryption. In encryption process first four words w(0 – 3) are used as key in the first round but for decryption last four words w(40 – 43) are used in the first round.

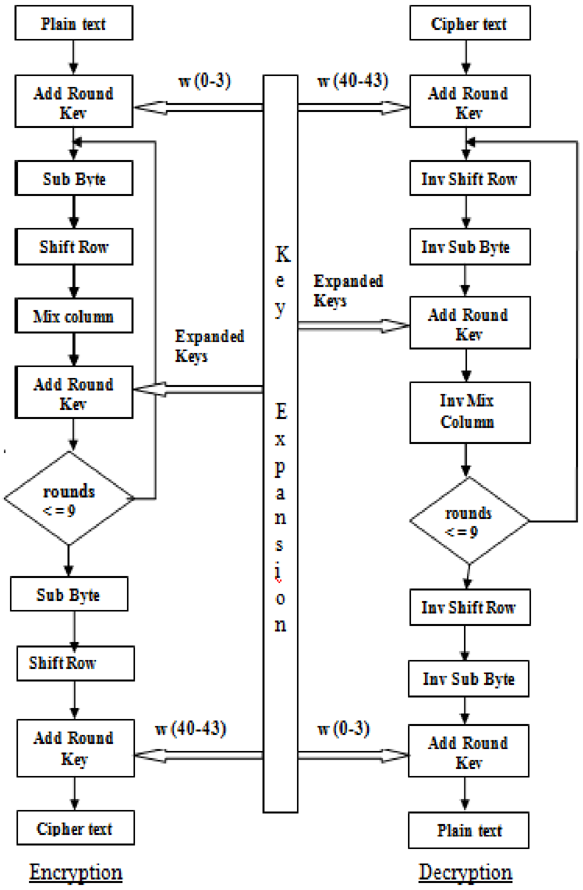

Fig 1 Structure of AES Encryption and Decryption algorithm

2. Related Work

Rijndael can also be implemented very efficiently on a wide range of processors and in hardware. Rafael R. Sevilla implemented on 80186 with assembly and Geoffrey Keating's Motorola 6805 implementation is also available on Rijndael site [14].

3. Implementation

3.1 Memory calculation:

The algorithm is implemented in C language. All variables with fixed value are assigned to program memory, whereas, all variables with variable values are assigned to RAM [8]. The memory required is calculated theoretically by noting the number of fixed and variable value variables in the algorithm.
ANSI C code is written for AES algorithm. This program is analyzed for RAM requirements. In the algorithm, the RAM is required for input key, input plain text, output

cipher value, state value, round keys, and temporary variables. Table 1 and 2 shows the RAM requirement of C code of AES algorithm. These tables show theoretical memory requirement for 10 rounds and 14 rounds respectively. As shown in tables, a uniform RAM is allocated for round keys.

Table 1: Theoretical Memory for AES with 10 rounds (key length = 128 bit)

Encryption		Decryption	
Variables	Memory values in Bytes	Variables	Memory values in Bytes
Key	16	Key	16
Temp variables	12	Temp variables	16
In	16	In	16
Out	16	Out	16
State	16	State	16
Round Key	240	Round Key	240
Total	316	Total	320

Table 2: Theoretical Memory for AES with 14 rounds (key length = 256 bit)

Encryption		Decryption	
Variables	Memory values in Bytes	Variables	Memory values in Bytes
Key	16	Key	16
Temp variables	16	Temp variables	20
In	16	In	16
Out	16	Out	16
State	16	State	16
Round Key	240	Round Key	240
Total	320	Total	324

3.2 Compilers:

The AES algorithm was implemented using ANSI C language. Two popular compilers KEIL and SDCC are used for compiling the algorithm coded in C language. The memory required by the algorithm in both cases is noted down from the complier result. The hex file is ported to target device 8051 microcontroller.

The KEIL compiler generally adds overheads for each feature. By default, KEIL keeps lot of features enabled and hence requires more RAM. The KEIL compiler allows using medium or large memory model for compiling AES algorithm, whereas SDCC allows using small memory model. Therefore, it expected that memory requirement of AES complied with SDCC will be less than that of the code compiled with KEIL. Hence, SDCC is recommended for achieving lowest possible memory for AES implementation on embedded systems.

To optimize the memory requirement some of the standard optimizations techniques incorporated in embedded C compilers are available on both the compliers [8]. For example, the SDCC uses the following techniques:

1. Global sub expression elimination.
2. Loop optimizations (like loop invariant, strength reduction of induction variables and loop reversing).
3. Constant folding and propagation, copy propagation.
4. Dead code elimination and jump tables for 'switch' statements.
5. Microcontroller architecture based specific optimizations, including a global register allocator.
6. Independent rule based peep hole optimization.

3.3 Results:

The AES algorithm is compiled by both compilers, viz., KEIL and SDCC. In each case, memory is measured in debug mode of the compiler. It gives out the total RAM memory of the compiled program.

Table 3 and Table 4 show required memory after compilation on different compilers taken for test condition. It is observed that memory requirement is entirely different for both KEIL and SDCC compiler. Also, the required memory value differs from theoretical value. The additional memory used by compilers is the overhead for generating the compiled code. It is also observed SDCC compiler occupy less memory space for this application compared to KEIL. However, KEIL tool is superior to SDCC in terms of debugging and code development speedup.

Table 3: Comparison of Memory for AES with 10 rounds (key length = 128 bit)

Method/Cross Compiler used	Encryption	Decryption
	Memory values in Bytes	Memory values in Bytes
Theoretical Values	316	320
KIEL Compiler	1378	1382
SDC Compiler	358	476

Table 4: Comparison of Memory for AES with 14 rounds (key length = 256 bit)

Method/Cross Compiler used	Encryption	Decryption
	Memory values in Bytes	Memory values in Bytes
Theoretical Values	320	324
KIEL Compiler	1382	1386
SDC Compiler	395	498

In order to demonstrate, an embedded board with P89V51RD2 microcontroller and 1024 RAM is selected. Since available RAM is 1024 bytes, the AES algorithm compiled by KEIL is not possible to port into this board. This is because the program size is 5748 bytes when compiled with KEIL. Whereas, the AES algorithm compiled with SDCC is ported successfully into the board since it requires a maximum RAM of 498 bytes. Table 5, shows the results of embedded board. As shown in the Table 5, the required code size is much smaller compared to that of KEIL.

Table 5: Execution time and code size of Rijndael AES on ATMEL 89V51RD2 compiled with SDCC.

Key/Block length	Number of Cycle	Code length
128/128	3168	728
256/128	5221	956

4. Conclusions

The objective of this paper is to study memory optimization for implementing the advanced encryption algorithm Rijndeal on embedded system. The theoretical and actual memory requirements are analyzed using two very popular compilers KEIL and SDCC and found that the actual memory required is more than the theoretical values in 10 rounds and 14 rounds for both 128 bit and 256 bit key length. This is mainly due to the availability of less number of registers on the microcontroller which forces compiler to use on chip RAM area. In this regard, SDCC is superior for realizing AES algorithm with minimum possible RAM. From Table 5, it is observed that execution time increases with number of rounds in AES.

References

[1] P.C. Van Oorschot A.J. Menezes and S. A. Vanstone, “Hand book of applied cryptography”, CRC Press, Waterloo, Ontario, Canada, 2001.

[2] Baker.W.“Introduction to analysis of Data Encryption Standard”, Laguna Hills CA; Aegean park press, 1991

[3] J. Daemen and V. Rijmen, AES Proposal: Rijndael (Version 2). NIST AES

[4] NIST, Advanced Encryption Standard (AES), (FIP PUB 197), November 26, 2001,

[5] Journal of research of the NIST, volume 106, November 3, May-June 2001

[6] M. McLoone, J. McCanny, “High Performance Single-Chip FPGA Rijndael Algorithm Implementations,” Proceedings Cryptographic Hardware and Embedded Systems Workshop, CHES, Paris, May 2001.

[7] T. Ichikawa, T. Kasuya, M. Matsui, “Hardware Evaluation of the AES Finalists,” in AES3:the third AES Candidate conference, New-York, April 13-14, 2000.

[8] B. Gladman, “The AES Algorithm (Rijndael) in C and C++, performance of the optimized implementation”. http://fp.gladman.plus.com/cryptographytech/rijndael/index.htm.

[9] Helion Technologies. High Performance (Rijndael) cores, 2001. Amphion Semiconductor. CS5210-40: High Performance AES Encryption Cores, 2001. http://www.amphion.com/cs5210.html

[10] A.J. Elbert, E. Yip, B. Chetwynd, C. Paar: “An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists”, IEEE Transactions on VLSI, August 2001, vol. 9, no. 4, pp. 545-557.

[11] Dai, Wei. “ Speed Comparison of Popular Crypto Algorithms. Performance of Crypto algorithms in software”, http://www.eskimo.com/~weidai/benchmarks.html (2001).

[12] “cryptography and Network security principles and practices” by William Stallings, Eastern economy edition publication 4th edition 2006

[13] Lomont, Chris. AES – Advanced Encryption Standard. Software performance of the Rijndael, http://www.math.purdue.edu/~clomont/software/AES/AES.htm (2001).

[14] Rijndael home site http://www.esat.kuleuven.ac.be/~rijmen/rijndael/

A.R.Reddy is professor and Head, the Department of Electronics and Communication Engineering, Madanapalle, Andhra Pradesh, India. He received M.Tech. and Ph.D. from IIT, Kharagpur, India in 1981 and 1985, respectively. He has conducted research work for 22 years at ITI Ltd, Bangalore, and developed communication products for Indian Army. He has published more than 50 research papers. His current research interests are cryptography and Embedded System and guided 2 Ph.D. students in the same area.

M. Ravindra Babu A received B.Tech. in Electronics and Communication Engineering from JNTUA Andhra Pradesh. He is pursuing M.Tech (digital Electronics and Communication System) at department of Electronics and Communication Engineering, Madanapalle Institute of Technology and Science, Madanapalle, Andhra Pradesh.