Parallelization of AES Algorithm Using OpenMP

S. S. Navalgund, Akshay Desai, Krishna Ankalgi, and Harish Yamanur
Department of Electronics and Communication Engineering, SDMCET, Dharwad, Karnataka, India
Email: siddunavalgund@yahoo.com, {akshaysd08, krishna.ankalgi, harishyamanur32}@gmail.com

***Abstract*—Advanced Encryption Standard (AES), a Federal Information Processing Standard (FIPS), is an approved cryptographic algorithm that can be used to protect electronic data. AES is a complex algorithm that requires large number of mathematical computations to be done. A sequential implementation would require a considerable amount of execution time. This may not be feasible for some applications that require faster rates of encryption and decryption to match the required data flow. This paper proposes an optimised parallel architecture of AES algorithm at both data and control level, suitable to be implemented in a multicore environment. The AES algorithm has been implemented in C language and is parallelised using OpenMP standard. The performance analysis is done using Intel VTune™ Amplifier XE 2013. The proposed parallel design exhibits improved performance over the sequential approach which addresses several applications that have a time constraint.**

***Index Terms*—AES, encryption, decryption, parallelization, multicore, OpenMP, VTune Amplifier**

I. Introduction

The importance of cryptography applied to security in electronic data transactions has acquired an essential relevance during the last few years. Each day millions of users generate and interchange large volumes of information in various fields, such as financial and legal files, bank services via Internet and e-commerce transactions. These and other examples of applications deserve a special treatment from the security point of view, not only in the transport of such information but also in its storage.

Advanced Encryption Standard (AES), also known as Rijndael, is a block cipher adopted as an encryption standard by the US government. This cipher was developed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen and submitted to the AES selection process under the name "Rijndael", a portmanteau comprised of the names of the inventors [1]. The AES algorithm is a symmetric block cipher that can encrypt and decrypt information thus providing a high level of security to the electronic data.

In addition to security level, the cipher speed is the most important feature of cryptographic algorithms. In this paper we propose a software approach based on transformations of a source code written in C language representing the sequential AES algorithm. The main contribution of this paper is to present the parallelization process of the AES algorithm along with the description of exploited parallelization methods and speed-up measurements.

The paper is organized as follows; the AES algorithm is briefly reviewed, a brief description of the parallelization tools that were utilized is given, which is followed by a brief explanation of the parallelization process of the AES algorithm. Lastly the experimental results regarding the application efficiency of the parallel algorithm are presented.

II. Description of AES Algorithm

AES is based on the principle known as Substitution Permutation network (SP-network) which means there will be a series of linked mathematical operations in the block cipher algorithm [2].AES encrypts a data block of 128-bits which is fixed with three different key sizes 128,192,256 bits. The operations are based on Rijndael algorithm. The input of AES algorithm is 128-bit or 16 byte data which can be specified as a block. The basic unit of processing in the AES algorithm is a byte. All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between the braces in the order ($b7$, $b6$, $b5$, $b4$, $b3$, $b2$, $b1$, $b0$). These bytes are interpreted as finite field elements using a polynomial representation as follows

$$b_7X_7+ b_6X_6+ b_5X_5+ b_4X_4+ b_3X_3+ b_2X_2+ b_1X_1+b_0 \quad (1)$$

Internally in AES algorithm operations are performed on a two-dimensional array of bytes called the state. The state consists of four rows of bytes, each containing Nb bytes, where Nb is the block length divided by 32 (4 for 128-bit key, 6 for 192-bit key, 8 for 256-bit key). Likewise the key length and number of rounds (iterations) differ from key to key as shown in Table I [3].

The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plain text into the final output of cipher text. Each round consists of several processing steps, including one that depends on the cipher key. A set of reverse rounds are applied to transform cipher text back into the original plain text using the same cipher key. AES does not use a Feistel network in the sense that the process of decryption slightly changes with respect to encryption. Fig. 1 represents the sequential flow of the algorithm [3].

High-level description of the algorithm is as follows,

A. Key Expansion

Round keys are derived from the cipher key using Rijndael key schedule

Manuscript received June 10, 2013; revised August 28, 2013.

doi: 10.12720/lnit.1.4.144-147

B. Initial Round

Add Round Key: In the add round key step the 128 bit data is XORed with the sub key of the current round using the key expansion operation. The add round key is used in two different places one during the start that is when round r=0 and then during the other rounds that is when 1 < round < Nr, where Nr is the maximum number of rounds.

C. Rounds

- Substitute Bytes: A non-linear substitution step where each byte is replaced with another according to a lookup table.
- Shift Rows: A transposition step where each row of the state is shifted cyclically a certain number of steps.
- Mix Columns: A mixing operation which operates on the columns of the state, combining the four bytes in each column.
- Add Round Key

D. Final Round

- Substitute Bytes
- Shift Rows
- Add Round Key

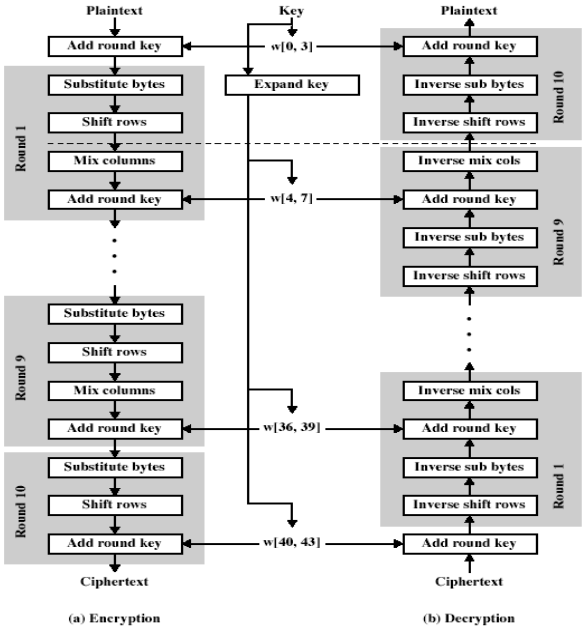

Figure 1. The sequential flow of AES algorithm.

TABLE I. DIFFERENT KEYS AND ITS ATTRIBUTES

Algorithm	Key Length (Nk words)	Block Size (Nb Words)	No.of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

III. PARALLELIZATION TOOLS

In order to parallelize the AES algorithm we have used OpenMP API directives to overcome data dependency and synchronization problems. Intel VTune™ Amplifier XE 2013 has been used to recognize that part of the code which consumes maximum computational time and then optimize it.

A. OpenMP API

The OpenMP Application Program Interface supports multi-platform shared memory parallel programming in C/C++ and Fortran on all architectures including Unix and Windows NT platforms[4]. OpenMP is a collection of compiler directives, library routines and environment variables that can be used to specify shared memory parallelism. OpenMP directives extend a sequential programming language with Single Program Multiple Data constructs, work-sharing constructs, synchronization constructs and make possible to operate with shared data and private data. An OpenMP program begins execution as a single task called master thread. When a parallel region is encountered, the master thread creates a team of threads. The statements within the parallel region are executed in parallel by each thread in the team[5]. At the end of the parallel region, the threads of the team are synchronized. Then again only the master thread continues execution until the next parallel region will be encountered. It is necessary to find remedy for all problems connected with programming restrictions on parallel processing to build a valid parallel code

B. VTune™ Amplifier XE 2013

The dynamic computation characteristics of the benchmarks are profiled and VTune ™ Amplifier XE 2013 from INTEL® has been utilized as the performance analyzer to identify the hotspots[6]. VTune analyzes the software performance on IA-32 and IA-64 based machines. It collects performance data on applications running on the host system, organizes and displays the data in an interactive way. VTune's call graph view provides a tree structure to show the call relationship among all functions along with their execution time. This would help us identify the "hotspot" functions and percentage of the hotspot functions occupying the total execution time of each benchmark.

IV. PARALLELIZATION OF AES USING OPENMP

In cryptography, the simplest mode of operation used with a block cipher to implement the complete encryption algorithm is the Electronic Code Book (ECB) mode [7]. The entire plain text is divided into blocks of fixed length which can be processed independently. Each block of plaintext is encrypted with the same key as a unit and turned into cipher text block.

The ECB mode of encryption is followed to implement the AES algorithm. Fig. 2 represents the ECB mode of encryption for AES algorithm. Blocks of length 128-bits are formed from the given plain text. The ECB mode of encryption supports a parallel architecture as the

individual blocks of plain text can be processed independently. The decryption of the cipher text can also be done in a similar way.

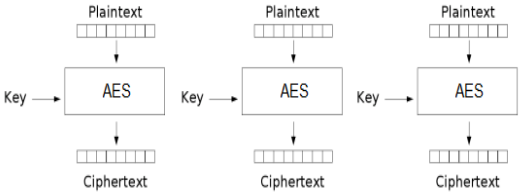

Figure 2. ECB mode of encryption for AES algorithm.

The sequential algorithm can be modified to take advantage of the multiprocessing units. According to the parallel computations paradigms[8], the independent parts of the algorithms must be identified and then prepared to work in separate threads. Initially the AES algorithm is divided into parallelizable and un-parallelizable parts. The parallelized portion and un-parallelizable portion are joined using the fork-join model.

The data input of the parallelization process is the well optimized sequential AES algorithm. The process of the AES algorithm parallelization can be divided into the following stages:

- Finding the most time-consuming functions of the AES algorithm using VTune Amplifier XE 2013.
- Making preliminary transformations of the most time consuming loops.
- Data dependences analysis of the most time consuming loops
- Constructing parallel loops in accordance with the OpenMP API
- Verification of a parallelized source code.

The result of the parallelization process is a parallelized AES algorithm which shows improved performance over a sequential implementation.

V. Results

TABLE II. Comparison of Execution Times of Sequential and Parallelized Code

File Size	Execution Time	
	Sequential Code	Parallelized Code
5 KB	0.045 sec	0.025 sec
10 KB	0.072 sec	0.058 sec
20 KB	0.128 sec	0.104 sec
40 KB	0.245 sec	0.211 sec
50 KB	0.334 sec	0.279 sec
100 KB	0.770 sec	0.698 sec
200 KB	1.343 sec	1.002 sec
400 KB	2.461 sec	1.867 sec
800 KB	5.439 sec	4.238 sec
1 MB	6.799 sec	5.010 sec
2 MB	13.649 sec	11.483 sec
5 MB	32.397 sec	27.380 sec

The AES algorithm has been successfully parallelized by using OpenMP API directives and is optimized using VTune™ Amplifier XE 2013. The specifications of the system used are: Intel X64 architecture, 2.3GHz CPU. Table II specifies the execution times of sequential and parallelized code for different file sizes. It can be inferred from the table that as the file size increases the parallelized code offers better performance. The execution time of sequential code is greater than the parallelized code in all the cases. When the file size is less than 5KB the difference is not distinguishable.

Another analysis is done by changing the number of available threads for encrypting the same file of size 5KB. The specifications are shown in Table III.

TABLE III. Execution Time of Parallelized Code Based on Number of Threads

For a file of size 5KB	
No. of Threads	Time taken
1	0.045 sec
2	0.037 sec
3	0.032 sec
4	0.026 sec

The analysis and optimization of the obtained results is done using VTune™ Amplifier XE 2013. Fig. 3 shows the CPU usage histogram that represents the breakdown of elapsed time. It visualizes what percentage of the wall time the specific number of CPUs were running simultaneously. This is obtained by performing the hotspot analysis using VTune™ Amplifier XE 2013. The file encrypted is of the size 50KB.

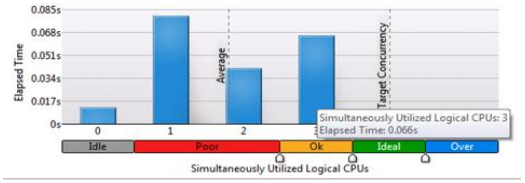

Figure 3. CPU usage histogram.

VI. Conclusion

In this paper, the parallelization of the AES algorithm is described. The AES algorithm was divided into parallelizable and un-parallelizable parts. OpenMP directives were used to parallelize the code by countering the problems of data dependency and synchronization. Then the concept of fork-join model was used to merge these parts to form a single piece of code. The comparison between execution times of sequential and parallelized codes shows considerable improvements after parallelization. The experiments carried out on the computer with multiple number of threads show that the application of the parallel AES algorithm considerably boosts the time of the data encryption and decryption. The results suggest that very attractive performance-effort ratios can be achieved by OpenMP based high-level language parallelization on modern symmetric multiprocessor platforms. The parallelized AES algorithm presented in this paper is also helpful for hardware implementations [9].

ACKNOWLEDGMENT

We are thankful to Department of ECE, SDMCET, for providing us with excellent laboratory facility. We also acknowledge with gratitude the contribution of journals, papers, organizations and books, which we have referred to in the list of the references.

REFERENCES

[1] J. Daemen and V. Rijmen, "AES proposal: Rijndael," *AES Algorithm Submission*, Sept 1999
[2] National Inst. of Standards and Technology, *Federal Information Processing Standard Publication 197, The Advanced Encryption Standard*, Nov 2001
[3] W. Stallings, *Cryptography and Network Security, Principles and Practices*, 4th ed. Pearson Education, 2006, pp. 134-161.
[4] OpenMP. [Online]. Available: http://www.openmp.org
[5] B. Chapman, G. Jost, and R. V. D. Pas, *Using OpenMP-Portable Shared Memory Parallel Programming*, MIT Press, Cambridge Massachusetts, London, England
[6] Intel Vtune. [Online]. Available: http://software.intel.com/en-us/intel-vtune/
[7] M. Dworkin, "Recommendation for block cipher modes of operation: Methods and techniques," *NIST Special Publication 800-38A*, December 2001.
[8] H. Kasahara and S. Narita, "Practical multiprocessor scheduling algorithms for efficient parallel processing," *IEEE Trans. Comput.*, vol. c-33, no. 11, pp. 1023-1029, Nov 1984.
[9] O-C. Mourad, S-M. Lotfy, M. Noureddine, B. Ahmed, and T. Camel, "AES embedded hardware implementation," in *Proc. 2nd NASA/ESA Conference on Adaptive Hardware and Systems*, 2007, pp.103-109.

Siddalingesh S Navalgund was born on 18th July 1978. He has completed Bachelor of Engineering (1999-2000) in Electronics and Communication engineering at SDM College of Engineering and Technology, Dharwad under Karnataka University Dharwad. He has completed M.Tech (2003-2004) at N.M.A.M.I.T under Visvesvaraya Technological University, Belgaum, Karnataka state, INDIA. He is presently pursuing PhD in the field of VLSI & DSP. He is currently working as Assistant Professor in Department of Electronics and Communication engineering in SDM College of Engineering and Technology, Dharwad. He has many papers at national and international level to his credit.

Mr. Navalgund is a life time member of ISTE, a member of Institution of Engineers (INDIA) and is also a member of IETE.

Akshay Desai was born on 8th May 1992. He is currently pursuing Undergraduate studies; in the final year of Bachelor of Engineering (2012-2013) in Electronics & Communication engineering at SDM College of Engineering and Technology, Dharwad under Visvesvaraya Technological University, Belgaum, Karnataka state, INDIA.
He has presented a paper on FPGA implementation of AES algorithm at the National Conference NCCTIA'12 and won the best paper award. His research interests include data security, parallel processing and systems, wireless communication.
Mr. Desai is a student member of Institution of Electronics and Telecommunication Engineers (IETE).

Krishna Ankalgi was born on 10th July 1991. He is currently pursuing Undergraduate studies, in the final year of Bachelor of Engineering (2012-2013) in Electronics & Communication engineering at SDM College of Engineering and Technology, Dharwad under Visvesvaraya Technological University, Belgaum, Karnataka state, INDIA.
He has presented a paper on FPGA implementation of AES algorithm at the National Conference NCCTIA'12 and won the best paper award. His research interests include data security, parallel processing and systems, VLSI.
Mr. Ankalgi is a student member of Institution of Electronics and Telecommunication Engineers (IETE).

Harish Yamanur was born on 17th October 1991. He is currently pursuing Undergraduate studies, in the final year of Bachelor of Engineering (2012-2013) in Electronics & Communication engineering at SDM College of Engineering and Technology, Dharwad under Visvesvaraya Technological University, Belgaum, Karnataka state, INDIA.
He has presented a paper on FPGA implementation of AES algorithm at the National Conference NCCTIA'12 and won the best paper award. His research interests include data security, parallel processing and systems, VHDL.
Mr. Yamanur is a student member of Institution of Electronics and Telecommunication Engineers (IETE).