International Journal of Software Engineering and Its Applications
Vol.8, No.2 (2014), pp.287-296
http://dx.doi.org/10.14257/ijseia.2014.8.2.28

Performance Improvement of Advanced Encryption Algorithm using Parallel Computation

M. Nagendra [#1] and M. Chandra Sekhar [#2]

[#]*Department of Computer Science & Technology,*
Sri Krishnadevaraya University, Anantapuramu- 515003, A.P, India
[1]*chandra.sm.gvp@gmail.com*

Abstract

The requirement of information security on network has become more and more important. Cryptography is a method to provide information confidentiality, authenticity and integrity. There are so many challenges to implement cryptography algorithm such as execution time, memory requirement, and computation power. Parallel computation is a promising technique to improve the performance of cryptography algorithm. Mainly divide-and-conquer strategy is used in parallel computation to solve the algorithms in parallel by partitioning and allocating, number of given subtask to available processing units. Parallel computation can be performed using multicore processors by parallelizing the execution of algorithm in multiple cores. In this paper we explore the implementation of AES (Advanced Encryption Algorithm) cryptography algorithm on dual core processor by using OpenMP API to reduce the execution time.

Keywords: *Cryptography, parallel computation, dual-core processor, OpenMP, AES*

1. Introduction

Now a days computer networks are becoming more important for exchanging information. One of the most important requirements of these networks is to provide secure transmission of information from one place to another. Cryptography is one of the techniques which provide most secure way to transfer the sensitive information from sender to intended receiver [2]. Its main purpose is to make sensitive information unreadable to all other except the intended receiver. So Advanced Encryption Algorithm is one of the most important cryptography algorithms for hiding the sensitive information. But AES algorithm has many performance limitations such as memory requirement and execution time [1].

So one of the solutions to reduce the execution time of AES algorithm is by using parallel computation. Parallel computation is a method in which several computations can be carried out simultaneously on two or more microprocessors. Parallel computation can be performed by using multicore and multiprocessor computers having multiple processing elements within a single machine. OpenMP (Open multiprocessing) is one of the application programming interface which is supported by multicore architectures to provide multithreaded shared memory parallelism. OpenMP uses fork join model for the parallel execution of code. Execution of OpenMP programs begin as a single process: the master thread. The master thread executes sequentially until the first parallel region construct is encountered and then it creates a team of parallel threads. So, statements in the program that are enclosed within the parallel region construct are then executed in parallel among the various threads. So by using multicore architectures we can parallelize the execution of AES algorithm among different cores to reduce the execution time of the algorithm [8].

ISSN: 1738-9984 IJSEIA

2. Advanced Encryption Algorithm

Advanced Encryption Standard is a symmetric key block-oriented cryptography algorithm. AES block cipher has 128, 192 or 256 bit key to encrypt or decrypt data in blocks of 128-bits [3]-[7]. Advanced Encryption Algorithm has a separate key expansion phase for the expansion of 128, 192 or 256-bit key so that these keys can be used in multiple rounds of encryption and decryption process.

Steps:

1) Input: Block size of 128 bit and 128, 192 or 256 bit key.
2) Perform simple bitwise XOR of the current block with the input key.
3) Perform the following steps 10, 12 and 14 times for 128,192 and 256 bit key respectively.
   - Substitute Byte step – Transformation by using non-linear byte substitution table (S-box) which operates on each of the bytes independently.
   - Shift row step - A simple permutation process that processes the State cyclically shifting the last three rows of the State by different offsets; Row 1 is circular left or right shift for encryption and decryption process respectively by one place, Row 2 by two, Row 3 by three places whereas, Row 0 remains unchanged [4].
   - MixColumns step - A substitution that make use of arithmetic over GF (28).
   - Add round Key Step – A simple bitwise XOR of the current block with a portion of extended key.
4) The key that is provided as input is expanded into an array of 44, 52 and 60, 32-bit words for 128,192 and 256 bit key respectively. This expanded key is used in each add round Key Step to perform bitwise XOR operation.
5) Repetition of step 3 for 10, 12 and 14 times for 128,192 and 256 bit key respectively will produce the cipher text as the output.

3. Parallel Implementation AES

Parallel implementation of Advanced Encryption Algorithm (AES) cryptography algorithm can be performed on multicore architectures by making the use of openMP API.

Multicore Architectures: A multicore system is a system which consists of two of more cores within a single processor. Here, core is nothing but a processing or execution unit. Multi-core architecture consists of two or more processing cores on the same chip. So, it is also referred as Chip Multiprocessor. In multi-core architecture design, each core has its own execution pipeline and each core has the resources required to run without blocking the resources needed by the other software threads.

Figure 1 shows the architecture of multi-core systems it has n number of processing cores integrated onto a single Chip. Each processing cores has its own private L1 cache and share a common L2 cache. The bandwidth between the L2 cache and main memory is shared by all the processing cores.

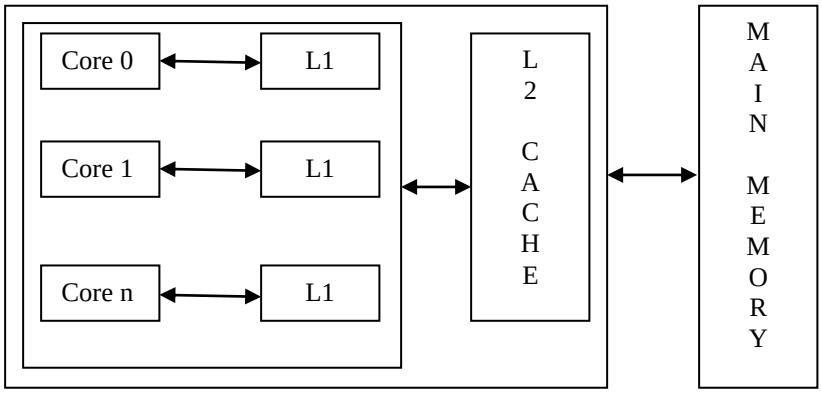

Figure 1. Architecture of Multicore System

The dual core processor contains two-cores, quad-core processor contains four cores and so on. Multi-core processor supports multiprocessing into a single physical package. Different cores in a multicore system can be coupled together loosely or tightly. Some of the common network topologies to interconnect cores are ring, bus, 2-dimensional mesh, crossbar etc. Multicore system supports the concept of simultaneous multithreading. Figure 2 shows the multicore system with simultaneous multithreading. It permits several independent threads to execute simultaneously on the same core. So, in multicore systems no. of threads, can execute multiple numbers of tasks simultaneously

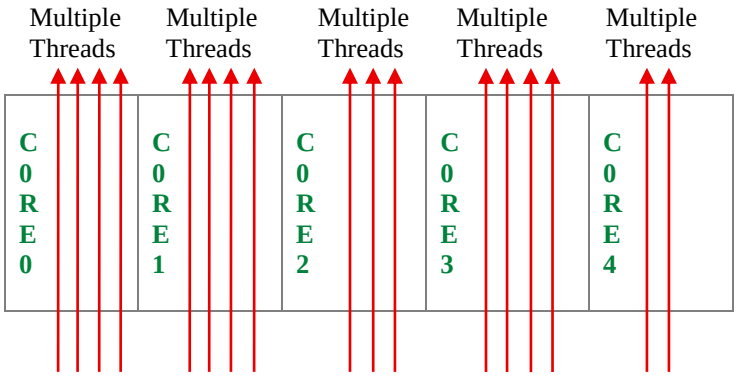

Figure 2. Multicore System with Simultaneous Multithreading

OpenMP

OpenMP (Open Multi-processing) is an application programming interface, it is jointly defined by a group of computer software and hardware vendors. OpenMP provides a scalable and portable model for developers of shared memory parallel applications. OpenMP supports programming languages C, C++ and FORTRAN on several architectures including Microsoft Windows and Unix platforms. Open MP uses fork-join model for the execution of any program or application. Fig3 shows the architecture of Fork-join model. All OpenMP program begins execution as a single thread of execution called the master thread. The master thread will executes in a single region until the first parallel construct is encountered. When a parallel construct is encountered then the master thread creates a team of parallel threads. The statements that are enclosed by the parallel region construct are then executed in parallel

among the various team threads. After the execution of all the statements within the parallel region, team threads will terminate and leaving only the master thread.

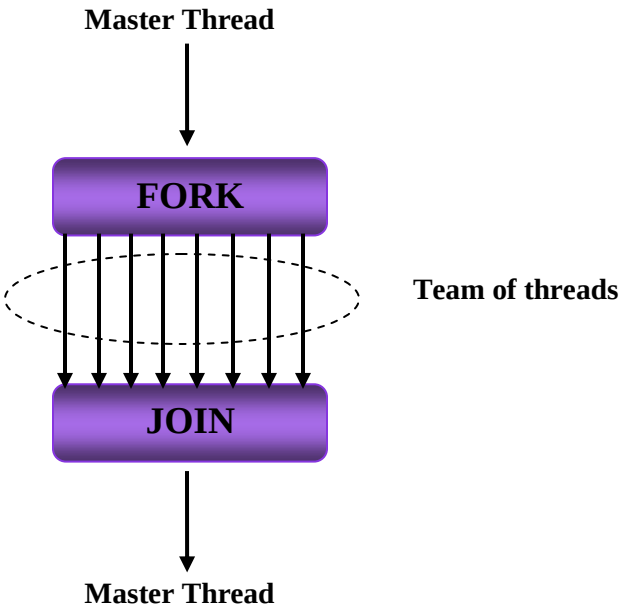

Figure 3. Fork-join Model

OpenMP mainly comprised of three components: Compiler directives, Runtime library routines and Environment variables. All Compiler directives are case sensitive. Some of the compiler directives are parallel region construct, section directive, single directive, barrier directive, master directive etc. Similarly some of the runtime library routines Omp_set_num_threads, Omp_get_num_threads, Omp_get_thread_num, Omp_get _num _ procs, Omp_get_max_threads *etc*.

OpenMP supports the various environment variables for controlling the execution of parallel code. Some of the environment variables are OMP_SCHEDULE, OMP_NUM_THREADS, OMP_DYNAMIC, OMP_NESTEDand OMP_STACKSIZE.

4. Implementation

The AES Cryptography algorithm has been implemented in dual core system by using OpenMP (Open multiprocessing) interface. Here we have parallelized the encryption and decryption process of AES cryptography algorithm by making the use of openMP directives, between the two cores to reduce the execution time. Figure 4 shows the flow chart for parallel implementation of encryption and decryption algorithm. File.txt is a text file for encryption/decryption, here its reading n-block of data at a time where n should be greater number such as 1000, 2000, 3000 *etc.*, to achieve the better performance. So that first n/2 blocks can be assigned to core-0 for encryption/decryption, while another n/2 bocks can be assigned to core-1 for performing encryption/decryption. In this case we are performing encryption/decryption on multiple blocks of data simultaneously by using the concept of simultaneous multithreading some of the bocks by core-0 and some of the blocks by core-1. This process will continue till the end of the file and after the last encryption/decryption step it will give the encryption/decryption time for entire file.

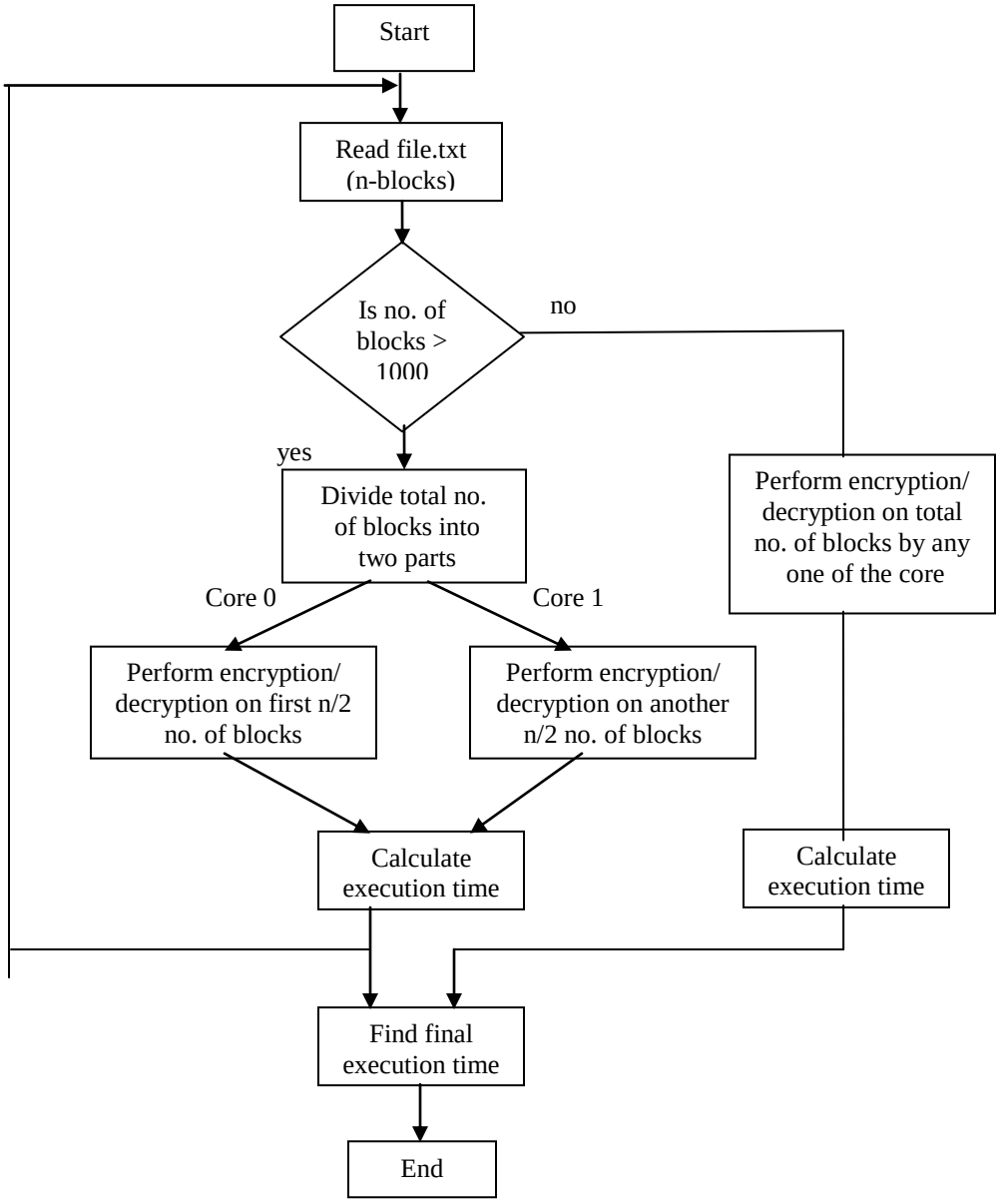

Figure 4. Flow Chart for Parallel Implementation

5. Results and Performance Analyses

The implementation results reported in this section makes the comparison between the sequential and parallel implementation of AES block cipher. In this paper, AES algorithm which is implemented by using the dual-core processor has been checked for correct execution time of encryption and decryption in both sequential and parallel implementation in the environment of Visual Studio 2005, intel C++ Compiler, Windows Xp, Core 2 Duo-2.4GHz, and 1GB RAM.

Table 1. Execution Time of AES algorithm for Encryption

Input File Size In KB	Time required for Encryption In seconds	
	Sequential Implementation	Parallel Implementation
1000	2.658211	1.60834
2000	5.316412	3.27653
3000	7.97529	4.917367
4000	10.632945	6.557235
5000	13.298456	8.197023
6000	16.98965	9.436251

Table 1 shows the execution time required by different size text files for encryption process. Here we reported two types of results. First of all, we show the execution time for different input plaintext size, in sequential implementation. Afterwards, we show the result of parallel implementation for same input plaintext size. Here it can be seen that implementation using parallel processing provides good performance. The mathematical formula to calculate the reduction in execution time (in %) for parallel implementation is as follows:

$$\text{Time} = \frac{\text{time in sequential imp. - time in parallel imp.}}{\text{time in sequential implementation}} \times 100 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

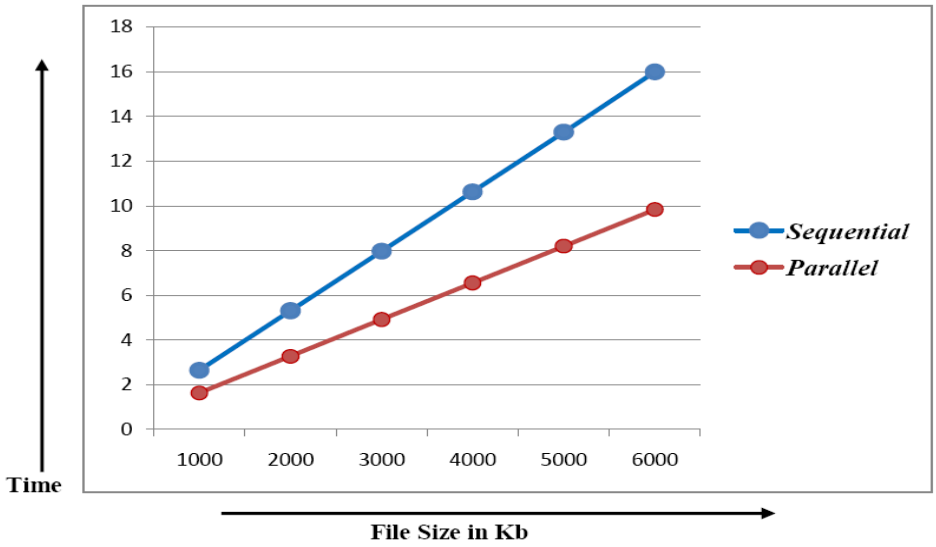

Figure 5. Execution Time for Encryption

Figure 5 shows graphical representation of time for encryption process. In this graph blue line shows the encryption time for sequential implementation and the red line shows the encryption time for parallel implementation. Graph shows the difference in execution time for sequential and parallel implementation for encryption process. Here we can see the performance improvement in the parallel implementation. In this it can be seen that the

performance is not fixed or constant for all file sizes. Here we reported that the performance of parallel implementation for small size file is less and it will increase as the file size will increase. But it will increase till a particular value and after that it will be a constant value. In our implementation value of reduction in time starts with 38% for very small size file and after that it will increase according to the file size but the maximum value for all the large files is 47% .

Table 2. Execution Time of AES Algorithm for Decryption

Input File Size In KB	Time required for Decryption In seconds	
	Sequential Implementation	Parallel Implementation
1000	5.46287	3.38923
2000	10.92637	6.773451
3000	16.387643	10.16564
4000	21.85154	13.56231
5000	27.36353	16.92865
6000	33.77453	19.13521

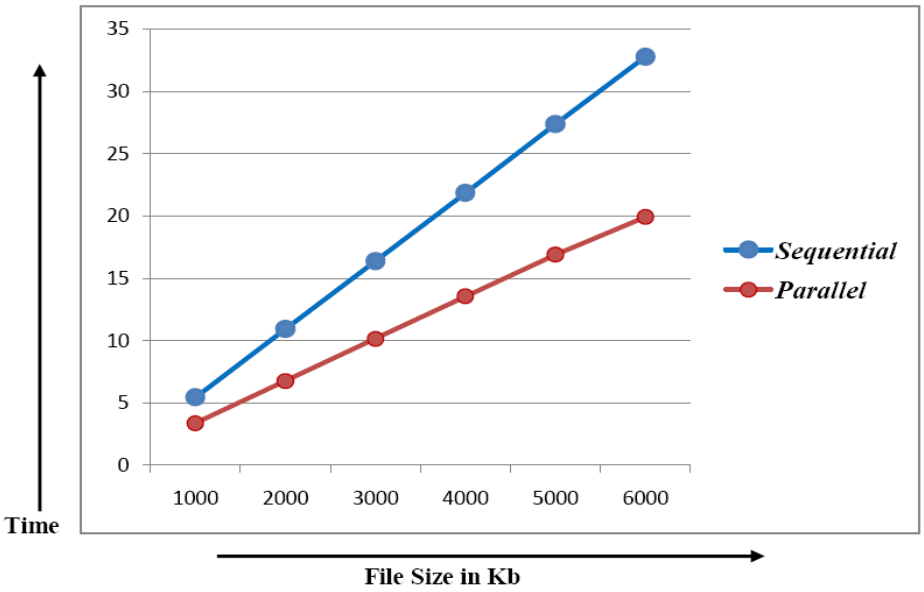

Figure 6. Execution Time for Decryption

Table 2 shows the execution time required by different size text files for decryption process. Here also we reported two types of results. First of all, we show the execution time for different input plaintext size, in sequential implementation. Afterwards, we show the result of parallel implementation for same input plaintext size. Here also it can be seen that implementation using parallel processing provides good performance. Above mentioned formulae can be used to calculate the reduction in time for parallel implementation.

Figure 6 shows graphical representation of time for decryption process. In this graph, blue line shows the decryption time for sequential implementation and the red line shows the decryption time for parallel implementation. Graph shows the difference in execution time for sequential and parallel implementation for decryption process. Here we can see the performance improvement in the parallel implementation. In this also it can be seen that the performance is not fixed or constant for all file sizes. Here we reported that the performance of parallel implementation for small size file is less and it will increase as the file size will increase. But it will increase till a particular value and after that it will be a constant value. In our implementation value of reduction in time starts with 38% for very small size file and after that it will increase according to the file size but the maximum value for all the large files is 45%.

On the basis of results of implementation we have described algorithm to calculate the value of encryption and decryption process for both sequential and parallel implementation.

Calculate time ()
{
Input: File size y in Kb
Output: Execution time for y kb file size in seconds.
Initialization:
x← 1, i←1, count←0;
Repeat while y!=0
$\delta_{i,0} \leftarrow$ y%10
$\delta_{i,1} \leftarrow$ x
y← y/10
x← x*10
count← count+1
i← i+1
end while

$$ET = c.[(\delta_{2,0} * \delta_{2,1})+\delta_{1,0}] \quad \text{for } 1 \leq y \leq 99 \tag{3}$$

$$ET = \sum_{n=3}^{count} \left(\sum_{z=1}^{\delta n,0} c.\, \delta n, 1 \right) \; c.[(\delta_{2,0} * \delta_{2,1})+\delta_{1,0}] \tag{4}$$

Where $y \geq 100$

}

The following algorithm takes file size of y kb as a input and it will calculate the execution time as the output. Here $\delta_{i,0}$ is a variable which stores the digits of input y, for *e.g.*, value of y is 45366 then the value of $\delta_{i,0}$'s will be $\delta_{1,0}$=6, $\delta_{2,0}$=6, $\delta_{3,0}$=3, $\delta_{4,0}$=5, $\delta_{6,0}$=4. $\delta_{i,1}$ is a variable which stores the numbers which are multiples of 10's. For $\delta_{1,1}$ it will store 1, similarly $\delta_{2,1}$=10, $\delta_{3,1}$=100, $\delta_{4,1}$ =1000, $\delta_{5,1}$ =10000 and so on. Count is a variable which counts the total number of digits in the input file size. For 45366 kb file size value of count is 5. After this equation (1) is given to calculate the execution time if the file size is less than or equal to 99 kb. While equation (2) can be used to calculate the execution time for a file if the file size is greater than or equal to 100 kb.

In both the equations value of c is constant and it depends upon the processor speed. The value of c varies from processor to processor. Here we have given the value of c for Core 2 Duo- 2.4GHz processor. From the implementation we have reported that value of c for encryption process in sequential implementation is $2.6582*10^{-03}$ and for parallel

implementation is $1.6394045*10^{-03}$. Similarly value of c for decryption process in sequential implementation is $5.4626465*10^{-03}$ and for parallel implementation is $3.38964*10^{-04}$
Here results of parallel implementation shows that the proposed system for implementation of AES cryptography algorithms using dual core architecture by using OpenMP application programming interface has been reduced the execution time for encryption and decryption processes. So by using parallel processing technique, we can improve the performance of the system.

6. Conclusion

In this paper, we have described concept of parallel programming by using multi-core processor. We have shown how to efficiently and effectively implement the Advanced Encryption Algorithm by using multicore systems and openMP API, extracting as much parallelism as possible from the algorithm in parallel implementation approach. We provided an extensive quantitative evaluation of execution time for both sequential and parallel implementation. After evaluation of execution time, we reported that parallel implementation of AES block cipher using dual-core (Intel Core 2 Duo) processor takes 40-45% less time for performing the encryption and decryption than the sequential implementation. These experiments allow us to confirm that, for the AES block cipher and similar algorithms, it is possible to efficiently use the multi-core processors for parallel implementation. Overall, we can conclude that multi-core processors provide an efficient and reliable way to implement AES cryptography algorithm.

References

[1] W. Liu, R. Luo and H. Yang, "Cryptography Overhead Evaluation and Analysis for Wireless Sensor", Networks Communications and Mobile Computing, 2009. WRI International Conference, vol. 3, **(2009)** January 6-8, pp. 496-501.
[2] W. Liu, B. Ying, H. Yang and H. Wang, "Accurate modeling for predicting cryptography overheads on awireless sensor nodes", Advanced Communication Technology. ICACT 2009. 11th International Conference, vol. 02, **(2009)** February 15-18, pp. 997-1001.
[3] M. B. Vishnu, S. K. Tiong, M. Zaini and S. P. Koh, "Security enhancement of digital motion image transmission using hybrid AES-DES algorithm", Communications, APCC 2008. 14th Asia-Pacific Conference, **(2008)**, pp. 1-5.
[4] C. Parikh and P. Patel, "Performance Evaluation of AES Algorithm on Various Development Platforms", Consumer Electronics, ISCE 2007. IEEE International Symposium, **(2007)**, pp. 1-6.
[5] A. M. Deshpande, M. S. Deshpande and D. N. Kayatanavar, "FPGA implementation of AES encryption and decryption", Control, Automation, Communication and Energy Conservation, INCACEC 2009, 2009 International Conference, **(2009)**, pp. 1-6.
[6] J. Yenuguvanilanka and O. Elkeelany, "Performance evaluation of hardware models of Advanced Encryption Standard (AES) algorithm", Southeastcon, IEEE, **(2008)**, pp. 222-225.
[7] F. Shao, Z. Chang and Y. Zhang, "AES Encryption Algorithm Based on the High Performance Computing of GPU", Communication Software and Networks, ICCSN '10. Second International Conference, **(2010)**, pp. 588-590.
[8] C.-F. Lu, Y.-S. Kao, H.-L. Chiang and C.-H. Yang, "Fast implementation of AES cryptographic algorithms in smart cards", Security Technology, Proceedings. IEEE 37th Annual 2003 International Carnahan Conference, **(2003)**, pp. 573-579.
[9] Q.-X. Zhu, L. Li, J. Liu and N. Xu, "The analysis and design of accounting information security system based on AES algorithm", Machine Learning and Cybernetics, International Conference, **(2009)**, pp. 57-62.
[10] B. Jyrwa and R. Paily, "An Area-Throughput Efficient FPGA Implementation of the Block Cipher AES Algorithm", Advances in Computing, Control, & Telecommunication Technologies, ACT '09. International Conference, pp. 328-332.
[11] D. Wang and X. Li, "Improved method to increase AES system speed", Electronic Measurement & Instruments, ICEMI '09. 9th International Conference, **(2009)**, pp. 3-49-3-52.

Authors

Dr. M. Nagendra, has obtained Phd., in 2008 from Sri Krishnadevaraya University. He is working as Associate Professor in Department of Computer Science & Technology. He has more than 18 years of teaching experience for both under graduate and Post graduate courses.

His areas of interests are Networking, High performance computing, Data communications, Artificial Intelligence, Programming languages and Cryptography. He has contributed more than 12 papers in various National and International journals. He is a Research Guide for nearly 8 scholars.

M. Chandra Sekhar, has obtained his M.Sc., (Computer Science), M.Sc (Stastistics) from Sri Krishnadevaraya University and M.Tech., in Computer Science from Acharya Nagarjuna Univeristy. At Present he is doing research on PhD., in Computer Science on Networks related topic High Performance Computing. He served as a Lecturer and Head of the Department in Sri Sai Degree & PG College, Anantapur for 8 years. Later he joined in IT industry and served as Software Engineer IBM and Accenture from past 7 years. His areas of interests are Networking, High Performance computing, SAP Netweaver, SAP BW. He has organized several workshops and training Programmes in the field of Computer Science and attended a number of Workshops and seminars. He is a life member of ISTE and Science & Society.