Improving Speed Up of Computing Using New AES Algorithm

Shweta Kumari[1], Abhishek Kumar[2]

[1, 2]School of Computing Science and Engineering, Galgotias University, Greater Noida, U.P, India

Abstract: ***With the improvement of cryptanalysis, applications are starting to use Advanced Encryption Standard instead of Data Encryption Standard progressively to protect their information security. However, current implementations of AES algorithm suffer from huge CPU resource consumption and low throughput. In this paper, we studied the technologies of parallel computing and its optimized design for encryption. Then, we proposed a new algorithm for AES parallel encryption, and designed and implemented a fast data encryption system. The test proves that our approach can accelerate the speed of AES encryption significantly.***

Keywords: Advanced Encryption Standard (AES); Speed-Up; Parallel Computing

1. Introduction

Most of the cryptographic algorithms like DES (Data Encryption Standard) or 3DES have some drawbacks when implemented in software: DES is no longer secure as computers get more powerful while 3DES is relatively sluggish in software. AES (Advanced Encryption Standard), which is rapidly being adopted worldwide, provides a better combination of performance and enhanced network security than DES or 3DES by being computationally more efficient than these earlier standards. Furthermore, by supporting large key sizes of 128, 192, and 256 bits, AES offers higher security against brute-force attacks. We will develop a new parallel AES algorithm for computing.

2. Advanced Encryption Standard

Advanced Encryption Standard (AES), has been proposed for replacing the previous encryption of Data Encryption Standard (DES) in 2001, it is considered as an alternative to DES. More and more applications are starting to use AES instead of DES to protect their information security in the past ten years [1].

Like DES, AES is a symmetric block cipher. This means that it uses the same key for both encryption and decryption. However, AES is quite different from DES in a number of ways. The algorithm Rijndael allows for a variety of block and key sizes and not just the 64 and 56 bits of DES' block and key size. The block and key can in fact be chosen independently from 128, 160, 192, 224, 256 bits and need not be the same [2]. However, the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128, 192, 256 bits. Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES- 256 respectively.

The AES parameters depend on the key length. For example, if the key size used is 128 then the number of rounds is 10 whereas it is 12 and 14 for 192 and 256 bits respectively. At present the most common key size likely to be used is the 128 bit key. This description of the AES algorithm therefore describes this particular implementation.

3. The AES Cipher

The Rijndael proposal for AES [3] defined a cipher in which the block length and the key length specified to be 128, 192, or 256 bits. The AES specification uses the same three key size alternatives but limits the block length to 128 bits. A number of AES parameters (Table1) depend on the key length. Most of the implementation of AES uses the key length of 128 bits.

Table 1: AES parameter

Key size (words/byte/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (word/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/178	52/208	60/240

4. Overall Structure of AES

The overall structure of AES is depicted in figure 1. The input to the encryption and decryption algorithms is a single 128-bit block [4]. This block of input is depicted as a square matrix of bytes. This block is copied into the state array, which is modified at each stage of encryption or decryption. After the final stage, state is copied to an output matrix. These operations are depicted in figure 2. Similarly, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and total key schedule is 44 words for the 128-bit key [5]. The ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the in matrix, the second four bytes occupy the second column, and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the w matrix [6].

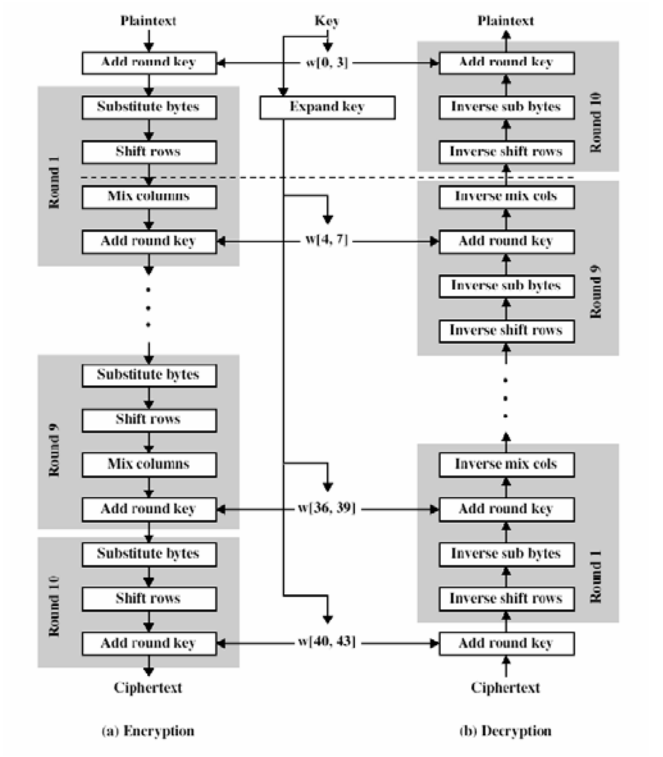


Figure 1: AES encryption and decryption

The four stages are as follows:
1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key
The tenth round simply leaves out the Mix Columns stage.

5. AES Key Expansion

The AES key expansion algorithm takes as input a 4-word key and produces a linear array of 44 words. Each round uses 4 of these words as shown in figure 2 [7]. Each word contains 32 bytes which means each subkey is 128 bits long. The most common way to attack block ciphers is to try various attacks on versions of the cipher with a reduced number of rounds. AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. As of 2004, the best known attacks are on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys [8]. Following is the flowchart of AES algorithm. The no of rounds depends on the key length specified, for 128, 192, or 256 bits it is 10, 12, and 14 respectively. The algorithm below is for key length 128 bits hence it contains 10 rounds:

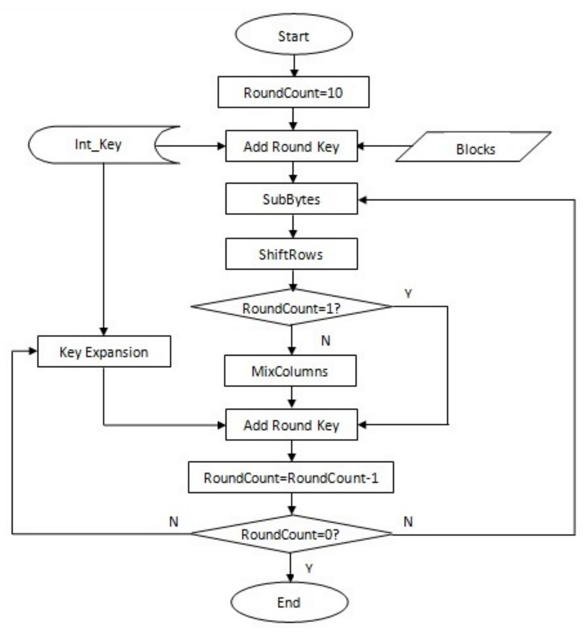


Figure 2: Flowchart of AES-128 algorithm

The AES cipher calculation is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext [9]. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform the ciphertext back into the original plaintext using the same encryption key. Figure 2 shows the flowchart of the AES-128 algorithm. From this figure, we can see that the AES-128 algorithm is iterative and consists of 10 rounds. The input is a block of data and the initial key. Each round operates on the intermediate result of the previous round and is a sequence of the four transformations, namely SubBytes, ShiftRows, MixColumns and AddRound-Key [10]. The intermediate result of any step is called the state. The final round is slightly different and the output after 10 rounds is the block of encrypted data.

6. Advance Encryption System for Parallel Computers

The current trend in high performance computing is clustering and distributed computing. In clusters, powerful low cost workstations and/or PCs are linked through fast communication interfaces to achieve high performance parallel computing [11]. Recent increases in communication speeds, microprocessor clock speeds, availability of high performance public domain software including operating system, compiler tools and message passing libraries, make cluster based computing appealing in terms of both high performance computing and cost effectiveness. The performance of a algorithm depends not only on input size but also on the architecture of the parallel computer, the number of processors, and the interconnection network [12].

There are two major components of algorithm design for parallel computers. The first one is the identification and specification of the overall problem as a set of tasks that can be performed concurrently. The second is the mapping of these tasks onto different processors so that the overall communication overhead is minimized [13]. The first component specifies concurrency, and the second one specifies data locality. The performance of an algorithm on a parallel architecture depends on both. Concurrency is necessary to keep the processors busy. Locality is important because it minimizes communication overhead. To implement the AES algorithm in parallel computers, data blocks (Figure 3) and a key are distributed among the available processors. Each processor will encrypt different data blocks using the same key. For example, in order to encrypt n number of data blocks with p processors, n/p data blocks will be encrypted by each processor. As each processor has its own data blocks and a key (increases data locality), all the 10/12/14 rounds (consists of four transformations) will be executed by each processor for encrypting each data block [14]. After encrypting all the data blocks of each processor, the encrypted data will be merged (Figure 4) in tree structure and return back to the main processor. For example, if there are four processors working in parallel, processor P1 will send its encrypted data to P0 and P0 will merge its encrypted data with P1; processor P3 will send its encrypted data to P2, and P2 will merge its encrypted data with P3. Finally processor P2 will send its (P2 & P3) encrypted data to P0 and P0 will merge its (P0 & P1) encrypted data with P2. This technique of merging and returning data to the main processor will increase the concurrency and reduce the idle time of each processor.

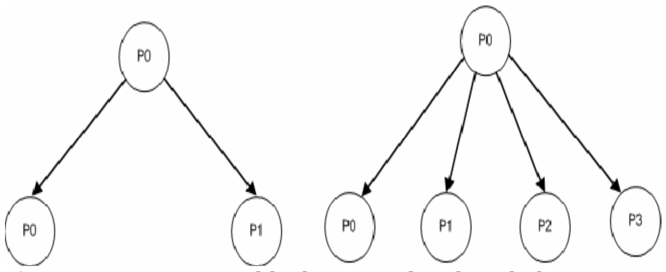


Figure 3: a) Data blocks are distributed between 2 processors
b) Data blocks are distributed among 4 processors

Decryption is done in the same way as the encryption. After decrypting the data blocks, the plain texts are merged and return back to the main processor in the same way as described above .

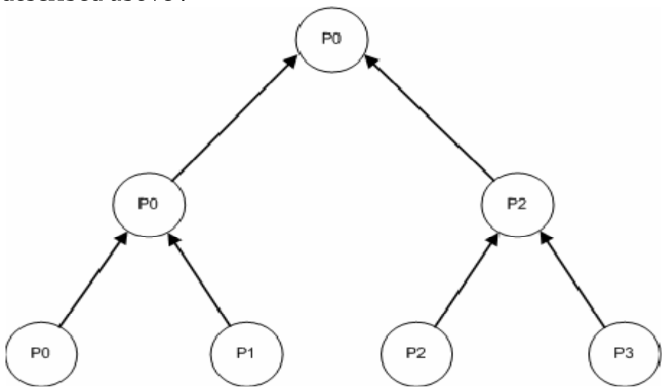


Figure 4: Encrypted data blocks are merged in tree structure

For parallel computing, the user can define threads which run in parallel using standard instructions we are familiar with within the field of general purpose programming [15]. The user declares the number of threads which must be run on a single SM by specifying a block size. The user also defines multiple blocks of threads by declaring a grid size. A grid of threads makes up a single kernel of work which can be sent to processors when finished, in its entirety, is sent back to the host and made available to the application.

7. Principle of AES Algorithm

In the traditional implementation of AES, the computation of data blocks is performed serially. Therefore, the efficiency and speed is poor. Figure 8 illustrates the principle of AES parallelism.

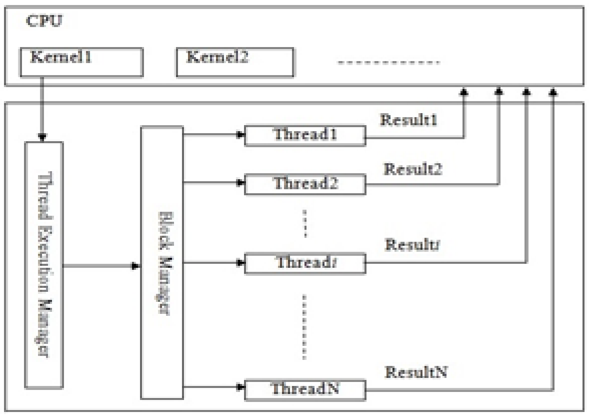


Figure 5: Principle of AES parallelism

From this figure, we can see that in parallel computers, it includes three components: the thread execution manager, the block manager and multiple threads. After CPU calls the kernel function the processing unit will enable the block manger active through the thread execution manager. The block manager will then divide plaintext into multiple blocks. Then, each block will be computed in individual thread. Finally, the encrypted block will be outputted to CPU, which will be assembled into the ciphertext [16]. Since parallel computing allows the number of thread in parallel to be the magnitude of one hundred thousand. Therefore, the AES encryption has high efficiency on parallel computing.

8. Modeling AES Algorithm for Parallel Computing

According to the principle of AES, we propose the AES algorithm for parallel computing. The flowchart is illustrated as follows:

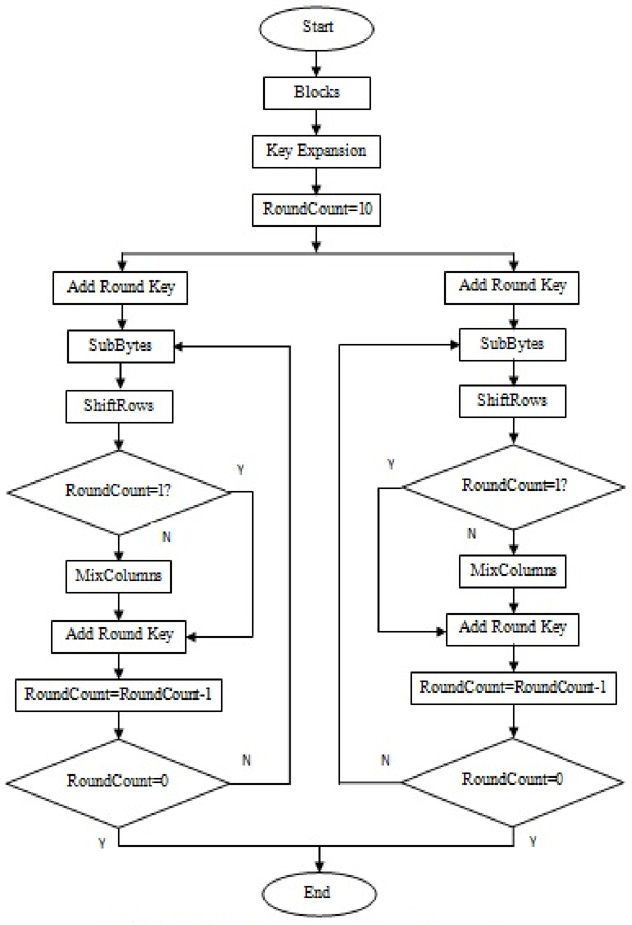


Figure 6: Flowchart of AES for Parallel Computing

According to the proposed AES algorithm for parallel computing in figure 6, we firstly store the plaintext and expanded key in the global memory space. The plaintext is then divided in blocks of 16 bytes which are encrypted completely in parallel. There are two general methods of AES computation [17]. One is the matrix computation, namely the 128-bit block is mapped into a 4*4 matrix. Then, the matrix is computed in a sequence of four stages: AddRoundKey, SubBytes, ShiftRows and MixColumns in each round. The other is the table lookup.

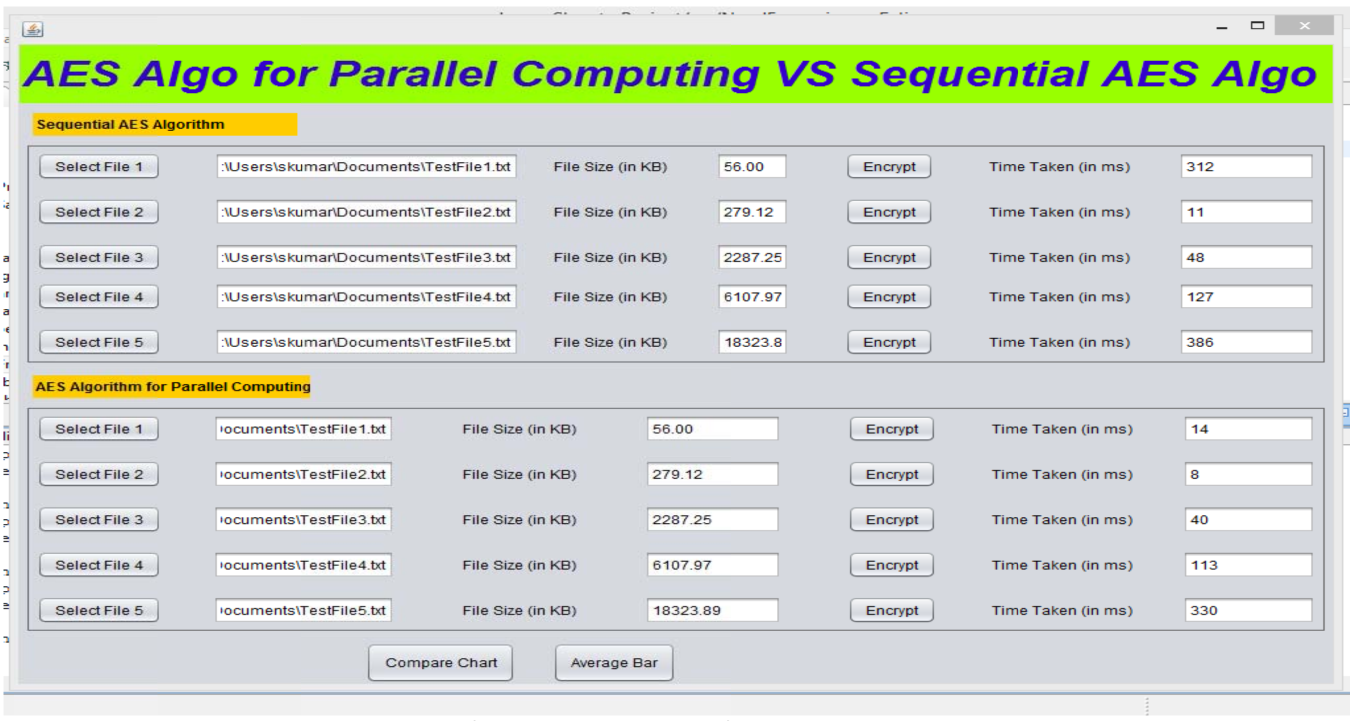
According to the proposed parallel AES algorithm, we firstly store the plaintext and expanded key in the global memory space. The plaintext is then divided in blocks of 16 bytes which are encrypted completely in parallel.

9. Performance Analysis and Evaluation

In this sub-section, we evaluate the performance of new AES algorithm for parallel computing that is used in a cipher system for data encryption. In our cipher system, the hardware is equipped with CPU of Intel Core 2 Duo E8200, the memory of 1GB. The software is our implementation of AES algorithm for parallel computing which runs in Windows XP. Tool used to achieve the goal is eclipse, and the language used is core java and swing.

A. Procedure Followed

Here we are showing the steps followed in order to achieve the better performance of parallel computing with the help of new AES algorithm. Figure 7 shows the window to select and compare the execution time between AES algorithm for parallel computing and sequential AES algorithm. The window contains two sections first one is for sequential AES algorithm and the second one is for AES algorithm for parallel computing. Left most column is to select file middle column will show the file size in kilobyte and after clicking the second right button of encrypt, the algorithm will encrypt the data and the time taken is shown in the rightmost column in millisecond.



AES Algo for Parallel Computing VS Sequential AES Algo

Sequential AES Algorithm

Select File 1	:\Users\skumar\Documents\TestFile1.txt	File Size (in KB)	56.00	Encrypt	Time Taken (in ms)	312
Select File 2	:\Users\skumar\Documents\TestFile2.txt	File Size (in KB)	279.12	Encrypt	Time Taken (in ms)	11
Select File 3	:\Users\skumar\Documents\TestFile3.txt	File Size (in KB)	2287.25	Encrypt	Time Taken (in ms)	48
Select File 4	:\Users\skumar\Documents\TestFile4.txt	File Size (in KB)	6107.97	Encrypt	Time Taken (in ms)	127
Select File 5	:\Users\skumar\Documents\TestFile5.txt	File Size (in KB)	18323.8	Encrypt	Time Taken (in ms)	386

AES Algorithm for Parallel Computing

Select File 1	›ocuments\TestFile1.txt	File Size (in KB)	56.00	Encrypt	Time Taken (in ms)	14
Select File 2	›ocuments\TestFile2.txt	File Size (in KB)	279.12	Encrypt	Time Taken (in ms)	8
Select File 3	›ocuments\TestFile3.txt	File Size (in KB)	2287.25	Encrypt	Time Taken (in ms)	40
Select File 4	›ocuments\TestFile4.txt	File Size (in KB)	6107.97	Encrypt	Time Taken (in ms)	113
Select File 5	›ocuments\TestFile5.txt	File Size (in KB)	18323.89	Encrypt	Time Taken (in ms)	330

Compare Chart   Average Bar

Figure 7: Time Taken for Encryption by AES for Sequential and Parallel Computing

After selecting the file and clicking the encrypt button we get the execution time, we will repeat same procedure for as much as file we want to select. Here we are selecting five files and the encryption time taken for all the five files are shown above, now the same process get repeated for AES algorithm for parallel computing.

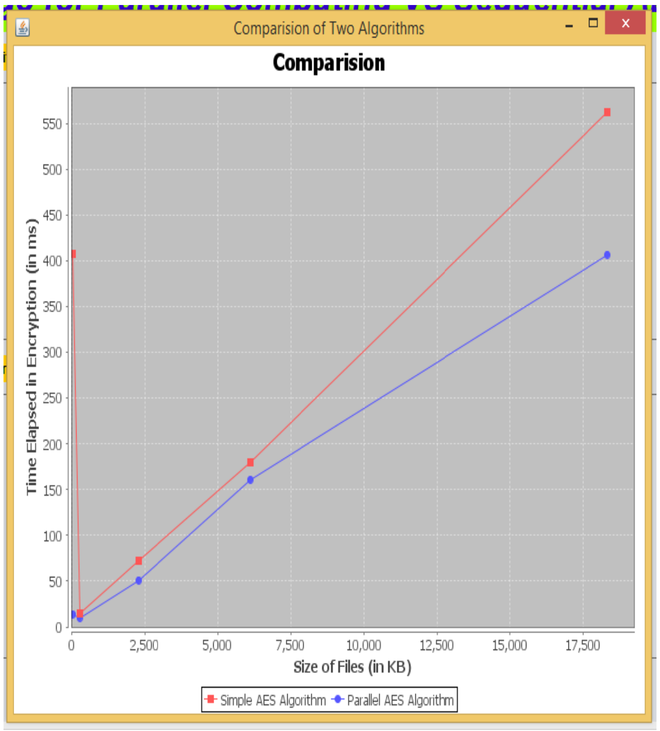


Figure 8: Comparison of Time Taken by Both Algorithms

Figure 8 illustrates the comparisons of AES encryption in terms of encryption time. In this figure, the horizontal axis indicates the size of files in terms of kilobyte (kb), the left vertical axis indicates the time in which the plaintext are encrypted into the ciphertext through AES algorithm in terms of millisecond (ms). Figure 9 is a bar graph of the overall comparison of both the algorithms for the five selected files:

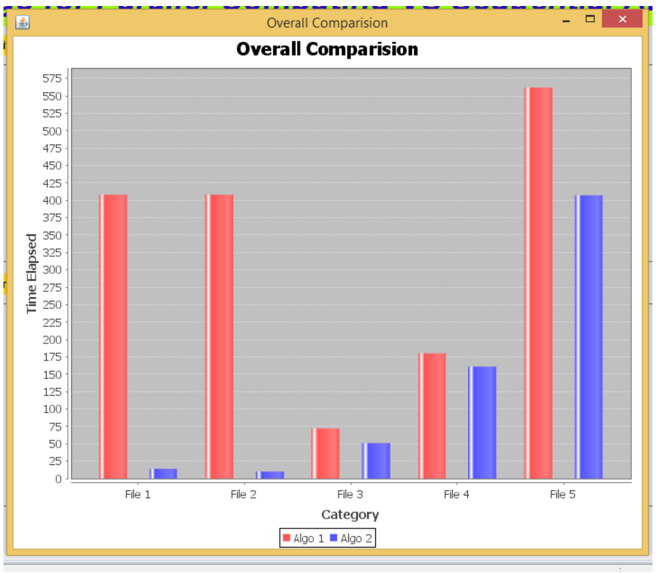


Figure 9: Overall Comparison of Both Algorithms

References

[1] N. Dunstan, "Semaphores for fair scheduling monitor conditions," SIGOPS Oper. Syst. Rev., 25(3):27–31,May 1991.

[2] M. Cole, "Algorithmic skeletons: structured management of parallel computation," MIT Press, Cambridge, MA, USA, 1991.

[3] L. Marziale, G. G. Richard III, and V. Roussev. Massive Threading: Using GPUs to Increase The Performance of Digital Forensics Tools. Digital Investigation, pp. S73-S81, 2007.

[4] Sarita V. Adve and Kourosh Gharachorloo, "Shared Memory Consistency Models: a Tutorial," IEEE Comput.,29(12):66-76, 1996.

[5] Ian Foster, "Designing and Building Parallel Programs," Addison Wesley, 1995.

[6] Murray I. Cole, "Algorithmic Skeletons: Structured Management of Parallel Computation," Pitman and MIT Press, 1989.

[7] Susanna Pelagatti, "Structured Development of Parallel Programs," Taylor & Francis, 1998.

[8] Kai Hong, "Advance Computer Architecture: Parallelism, Scalability and Programming," TMH, Edition 2001.

[9] J. Darlington, A. J. Field, P. G. Harrison, P. H. B. Kelly, D. W. N. Sharp, and Q. Wu, "Parallel Programming Using Skeleton Functions," In Proc. Conf. Parallel Architectures and Languages Europe, pages 146-160. Springer LNCS 694, 1993.

[10] National Bureau of Standards, NBS FIPS PUB 46, "Data Encryption Standard,", U.S. Department of Commerce, January 1977.

[11] J. Daemen, V. Rijmen. The Design of Rijndael: AES The Advanced Encryption Standard. New York, USA: Springer-Verlag, 2002.

[12] Daemon, J. and Rijmen, V. "The Rijndael Block Cipher: AES Proposal", NIST, Version 2, March 1999.

[13] Stallings, W. "Cryptography and Network Security: Principles and Practices." Third Edition, Pearson Education, Inc. 2003.

[14] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. In Proc. 2nd AES candidate conference, pp 15–34, NIST, 1999.

[15] J D. Owens, D. Luebke, N. Govindaraju et al. A Survey of General- Purpose Computation on Graphics Hardware. Computer Graphics Forum, Vol. 26, No. 1, pp. 80-113, 2007.

[16] RSA Laboratories, "The RSA Laboratories Secret-key Challenge: Cryptographic Challenges."

[17] Jaspal Subhlok, James M. Stichnoth, David R O'Hallaron, Thomas Gross, "Exploiting task and data parallelism on multicomputer," PPOPP '93 Proceedings of the fourth ACM SIGPLAN symposium on Principles and practice of parallel programming, ACM, NY, USA, 1993.