International Journal on Emerging Technologies 11(4): 347-351(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

Distributed Implementation of Efficient Symmetric Key Cryptic Algorithm of AES Algorithm using Multi Nodes

Shivlal Mewada[1], Sita Sharan Gautam[2] and Pradeep Sharma[3]
[1]Department of Physical Sciences, Mahatma Gandhi Chitrakoot Gramodaya Vishwavidyalaya, Chitrakoot, Satna (Madhya Pradesh) India.
[2]Department of Physical Sciences, Mahatma Gandhi Chitrakoot Gramodaya Vishwavidyalaya, Chitrakoot, Satna (Madhya Pradesh) India.
[3]Department of Computer Science, Govt. Holkar [Model, Autonomous] Science College, Indore India.

(Corresponding author: Shivlal Mewada)
(Received 14 May 2020, Revised 23 June 2020, Accepted 02 July 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: In the development of information technology, protecting sensitive information via cryptography methods is becoming more and more important in daily life. A cryptography method plays a dynamic role in providing information security against unauthorized users. That is, this technique increases the protection of the user password from unauthorized users. There are several issues and challenges to implementing and analyze of AES algorithm. Distributed computing is a promising method to enhance security and increase the efficiency of the AES algorithm. Distributed computation can be performed using multi-nodes by distrusting the execution of the algorithm in multiple cores.
The study present a distributed implementation and performance analysis of the AES algorithm using multi-nodes (Node-1, Node-2, Node-3) to reduce the execution time of the AES algorithm and compare the distributed exaction time with sequential exaction time of AES. The experiments show that the proposed AES distributed implementation is significantly better than the AES sequential implementation.

Keywords: Node-1, Node-2, Node-3 AES Algorithm, Cryptography.

I. INTRODUCTION

Information security on database server is very challenging specially while ensuring individual privacy of information contents.SKC techniques are applied to ensure sensitive information over the network [1]. Its main purpose is to prevent from the intended receiver. In 2001 replaces the DES or AES. AES can be used for online applications on variety of digital contents, like video/audio, and smart cards data. One of the main advantage of Rijndael algorithm is that it can be used for both hardware and software implementation. AES algorithm is a SKC algorithm. AES algorithms has many performance improvement hints such as memory requirement and execution time [1-4]. In order to improvise the execution time of AES algorithm is by using distributing computing technique. The Distributed computing environment is a widely-used industry standard that supports this kind of distributed computing. Several nodes in isolation are working on a single problem. In distributed computing, it is divided into many parts, and allocated to different nodes and can communicate with each other to achieve the gain.
The main objective of this paper to increase the efficiency and performance of AES algorithm using distributed computing. In this paper, we presents a distributed implementation and performance analysis of AES algorithm using multi-nodes (Node-1, Node-2, Node-3) to reduce the execution time of AES algorithm and compare the distributed exaction time with sequential exaction time of AES. The experiments show that the proposed AES distributed implementation is significantly better than the AES sequential implementation.
There search paper is categorized as follows, Section I contains the introduction of AES Algorithm with main objective of this paper, Section II contain the background of SKC algorithms of AES, Section III contain the methodology experimental set up, Section IV describes results and discussion of this study, Section V concludes research work with future directions.

II. AES ALGORITHM

AES algorithm accepts input block sizes 128, 192 and 256 bits uses key size - 128,192 and 256 bits. It depends on the length of key e.g. 10 iterations for 128 bit key, 12 iterations for 192-bit key and 14 iterations for 256 bit keys. In this algorithm, plain text transformed into cipher text after passing through the different stages like- byte substitution, row shift, column and iteration key [3-9].

Table 1: Key length and number of iteration of AES [1, 3-9].

Key Lengths (bits)	# Iterations = nr
128 bit	10
192 bit	12
256 bit	14

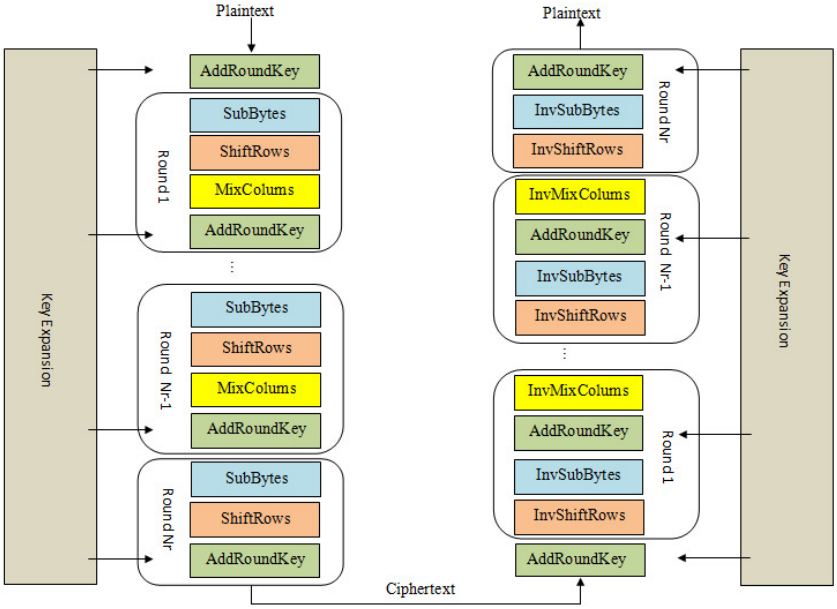


Fig. 1. AES Encryption and Decryption Process [1, 3-9]

Steps:
Encrypt_block(plain_text [16], cipher_text[16], array_round_key[R+1])
begin
block[16];
block = plain_text;
AddRoundKey(block, round_key[0]);
fori = 1 to R-1 step 1 to 4 do
step 1: SubBytes(block);
step 2: ShiftRows(block);
step 3: MixColumns(block);
step 4: AddRoundKey(block, round_key[i]);
end for
SubBytes(block);
ShiftRows(block);
AddRoundKey(block , round_key[R]);
cipher_text=block;
end

III. PROPOSED FRAMEWORK FOR DISTRIBUTED IMPLEMENTATION OF AES

Here, we have applied the distrusted algorithm in order to enhance the efficiency of AES using different nodes.

A. Steps of Distributed Implementation of AES Algorithm

1. Start by talking the file to be encrypted with encryption to be used
2. Divide the file into data block of 128 bits. If exact multipath of 128 bits is not possible and padding i.e. stuffing of bits is done to make a perfect block of 128 bits
3. Then take the block and select the version of AES to be used on the basis of which number of process round will be selected.
4. Then check availability of the processing node in the distributed environment on the basis if availability provides the block to the node.
5. Perform the encryption process on the basis of pre-defined number of round and number of available computing node.
6. The final integrate the results and store the time taken by the process.
7. Display the cipher file and stop.

With the increasing trends of utilizing more number of cores users are having their information on various devices as a decentralized information ion set. The secure exchange of this information and sharing computational resources can be utilized in the enciphering and deciphering process. This section performs encryption and decryption of data files of different formats of different sizes with fixed keys over a homogeneous distributed environment. The encryption and decryption task is realized using Hadoop (Fig. 2) distributed over a decentralized algorithm is implemented for effectiveness measurement. The results of encryption time, decryption time and Throughput have been depicted in graphical form as well as the tabular form to enhance the effectiveness.

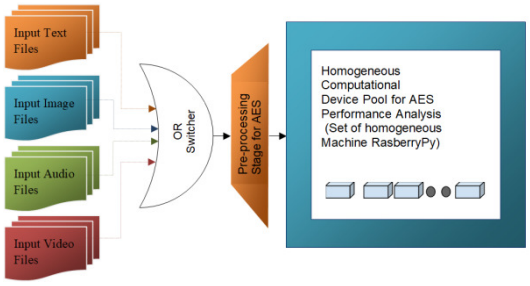


Fig. 2. Distributed AES file analysis framework using Hadoop.

B. Experimental Setup

All the sequential and distributed versions have been measured on different nodes, comprising performance analysis was run under windows 10 operating system with 16 GB of RAM available using three different Processors as following:

— Processor Intel(R) Core(TM) i5, 2 Core(s)
— Processor Intel(R) Core(TM) i5, 4 Core(s)
— Processor Intel(R) Core(TM) i5, 6 Core(s)
— Input Type: Text

— File Sizes: 8KB, 41KB, 72KB, 121KB

IV. IMPLEMENTATION AND PERFORMANCE ANALYSIS

For analysis of various input files with variable size and variation in version of encryption and decryption algorithms has been presented in Fig. 2. As depicted in figure, the input to AES algorithm is consist of purely single type of format either in static contents like text, image etc. The input files are made compatible to AES encryption/decryption process using preprocessing stage, whose function is to segment input files in appropriate block size (128bit,192bit, 256bit) depending upon the version of AES algorithm. The Hadoop Layer distributes the task among RasberryPy machines, and collects the statistics for analysis purpose. Table 2(a-c) offers 128 bit, 192 bit and 256 bit encryption time on node-1, node-2, and node-3 for text document file format. The respective graphical visualization has been presented in Fig. 3.

Table 2a: Encryption time of variants of AES128 bits with respect to variable file size.

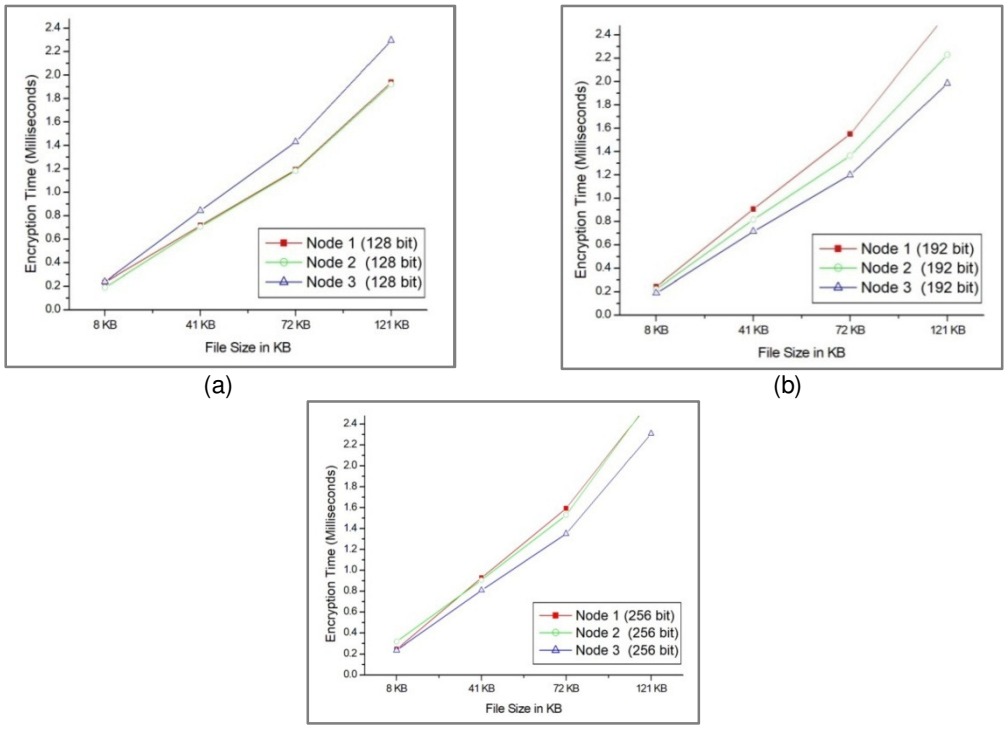
File Size in KB	128 Bit AES Algorithm Encryption Time (Milliseconds)		
	Node 1	Node 2	Node 3
8 KB	0.2334588	0.189516	0.2380161
41 KB	0.7171316	0.7065535	0.8440381
72 KB	1.1927315	1.1837054	1.4304351
121 KB	1.9399308	1.9229778	2.2948744

Table 2b: Encryption time of variants of AES 192 bits with respect to variable file size data.

File Size in KB	192 Bit AES Algorithm Encryption Time (Milliseconds)		
	Node 1	Node 2	Node 3
8 KB	0.2439327	0.2208806	0.1887451
41 KB	0.9071685	0.8166214	0.7141003
72 KB	1.5502999	1.3646433	1.1994399
121 KB	2.5878904	2.2287128	1.9857228

Table 2c: Encryption time of variants of AES 256 bits with respect to variable file size data.

File Size in KB	256 Bit AES Algorithm Encryption Time (Milliseconds)		
	Node 1	Node 2	Node 3
8 KB	0.2464692	0.3165006	0.235964
41 KB	0.9291581	0.9067466	0.8087673
72 KB	1.5949094	1.5300238	1.3519402
121 KB	2.6227136	2.6409323	2.3071556



(a) (b)

(c)

Fig. 3. Encryption time of with respect to variable file size on different no. of nodes.

Table 3a: Decryption time of variants of AES 128 bits with respect to variable file size.

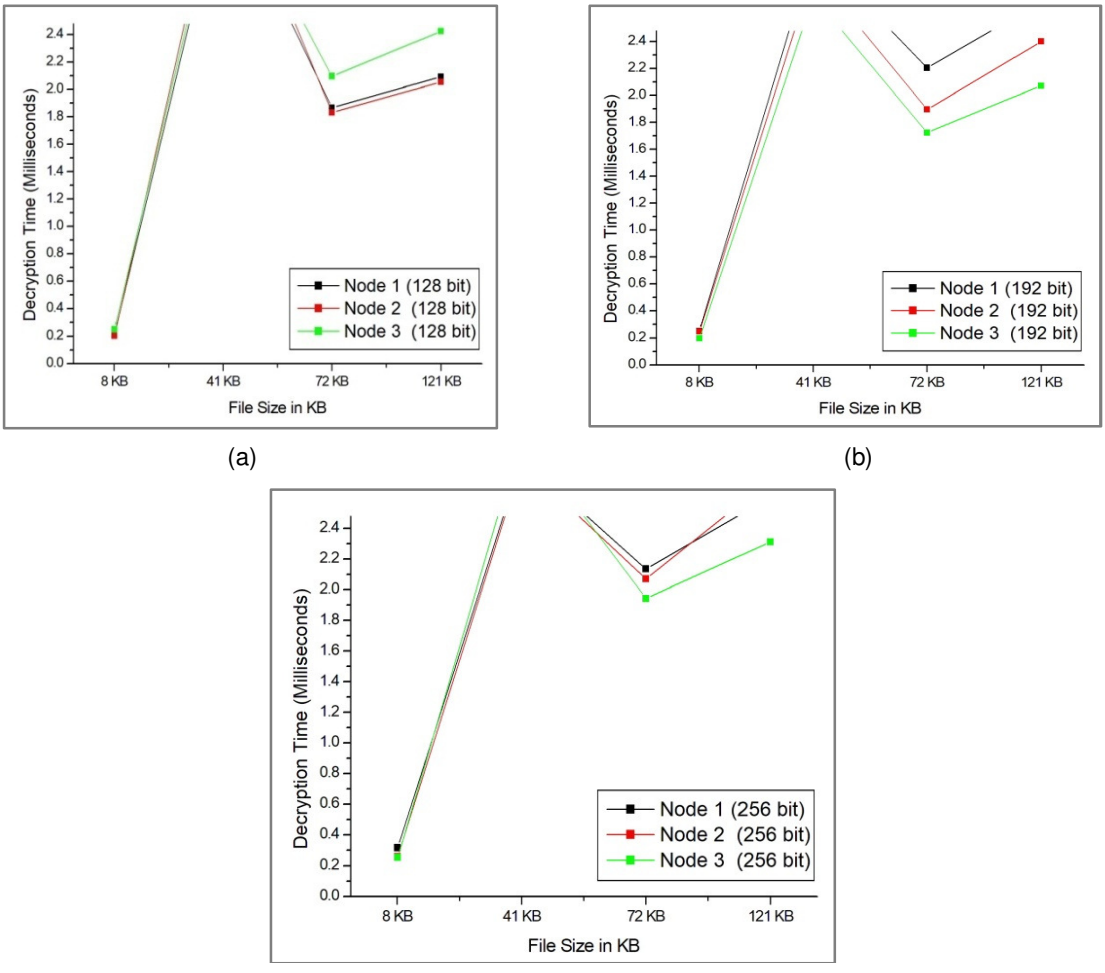
File Size in KB	128 Bit AES Algorithm Decryption Time (Milliseconds)		
	Node 1	Node 2	Node 3
8 KB	0.2131463	0.2031618	0.2499067
41 KB	3.519882	3.7410648	3.602658
72 KB	1.867561	1.8313682	2.0961187
121 KB	2.0914729	2.054452	2.4250307

Table 3b: Decryption time of variants of AES 192 bits with respect to variable file size.

File Size in KB	192 Bit AES Algorithm Decryption Time (Milliseconds)		
	Node 1	Node 2	Node 3
8 KB	0.2523078	0.2487817	0.1979118
41 KB	3.0790217	2.9230473	2.735396
72 KB	2.2045825	1.897535	1.7249304
121 KB	2.6904273	2.4004421	2.0711603

Table 3c: Decryption time of variants of AES 256 bits with respect to variable file size.

File Size in KB	256 Bit AES Algorithm Decryption Time (Milliseconds)		
	Node 1	Node 2	Node 3
8 KB	0.3148548	0.2593651	0.2541463
41 KB	2.8602295	2.8290212	3.015084
72 KB	2.1346345	2.0694675	1.9434674
121 KB	2.602651	2.715495	2.3108432



(a) (b)

(c)

Fig. 4. Decryption time of with respect to variable file size for Text files data on different no. of nodes.

The response of experimentation Table 2(a-c) provides encryption time on node-1, node-2 and node-3 for textual document file format. The respective graphical visualization has been presented in Fig. 3. As observation is made that with increase in the number of nodes higher encryption time is recorded. Now the response of experimentation Table 3(a-c) provides decryption time on node-1, node-2, and node-3 for audio document file format. The respective graphical visualization has been presented in figure.4 that shows a zig-zag pattern. But in general, there is overall decrement in decryption time.

The comparison of results obtained during the experiments is displayed in Table 2(a-c). Table 2(a-c) illustrates the effects of parameter like execution time, computing time, size of input text files for encryption process. Same is depicted for same parameters but for decryption process in Table 3 (a-c). Experimental results for execution computing time of variants of 0AES (128/192/256 bits) on different input plaintext size. Further observation of distributed implementation for same input plaintext size using 2 & 3 Node.

Fig. 3(a-c) & 4(a-c) shows graphically execution time for encryption and decryption process respectively, for sequential and distributed implementation for encryption process. One can see the performance improvement in the distributed implementation, which is not fixed or constant for all file sizes. However the performance of distributed implementation for small file is less over larger file.

Here experiment results of proposed AES algorithms using 2 and 3 node respectively for distributed computing has been reduced. So in distributed technique, the performance of the system has been increased.

Overall observation: On the basis of the computing time noted from the above tables, it is observed that with different variation of AES and multiple node configurations, approximately 18% enhancement is observed when 2 nodes with 128 bits AES is used.

V. CONCLUSION

AES systems with distributed algorithms for block cipher using node-2 with 128 bit takes 18% less time over the sequential implementation. These experiments demonstrates the efficiently usage of the multi nodes for distributed implementation. Thus multi-nodes version provides an efficient and reliable way to implement AES cryptography algorithm.

REFERENCES

[1]. M. Nagendra, M. & Chandra Sekhar (2016). *Performance Improvement of Advanced Encryption Algorithm using Parallel Computation.International Journal of Software Engineering and Its Applications, 8*(2) 287-296.

[2]. Ashok Sharma, Ramjeevan Thakur, Shailesh Jaloree, "*Investigation of Efficient cryptic Algorithm for cloud storage. Fourth International Conference on Recent Trends in Communication and Computer Networks*, India, 23-30.

[3]. Dimitrios Zissis, Dimitrios Lekkas (2012). *Addressing cloud computing security issues. Future Generation Computer Systems, 28*, 583–592.

[4]. Vivek Raich, Pradeep Sharma, Shivlal Mewada & Makhan Kumbhkar (2013). Performance Improvement of Software as a Service and Platform as a Service in Cloud Computing Solution. *International Journal of Scientific Research in Computer Science and Engineering*, *1*(6), 13-16.

[5]. Mewada, S., Sharma, P. & Gautam, S. S. (2016). Classification of Efficient Symmetric Key Cryptography Algorithms. *International Journal of Computer Science and Information Security*, *14*(2), 105-110.

[6]. Mewada, S., Sharma, P. & Gautam, S. S. (2016). Exploration of efficient symmetric AES algorithm. *IEEE Symposium on Colossal Data Analysis and Networking (CDAN),* 1-5. DOI: 10.1109/CDAN.2016.7570921

[7]. S. Mewada, P. Sharma and S. S. Gautam (2016). Exploration of efficient symmetric algorithms. *IEEE 3rd International Conference on Computing for Sustainable Global Development (INDIACom),* New Delhi, India, 663-666.

[8]. S. Mewada, S. S. Gautam, and P. Sharma (2016). Investigation of Efficient Cryptic Algorithm for Text using SMCrypter. *International Journal of Information Science and Computing, 3*(2), 99-108.

[9]. Mewada, S., Sharma, P. & Gautam, S. S. (2019). Investigation of Efficient SKC Cryptic Algorithm for Image Encipherment and Decipherment Using SMCrypter. *International Journal of Computer Sciences and Engineering*, *7*(4), 1220-1226.

How to cite this article: Mewada, S., Gautam, S. S. and Sharma, P. (2020). Distributed Implementation of Efficient Symmetric Key Cryptic Algorithm of AES Algorithm using Multi Nodes. *International Journal on Emerging Technologies, 11*(4): 347–351.