Journal of Science, Computing and Engineering Research (JSCER)
Volume-7, Issue-9, September 2024.
DOI: *https://doi.org/10.46379/jscer.2024.0709*
ISSN: 2708-1079

Performance Analysis of AES Techniques of Serial Implementation Algorithms

Jushak Prodhan, Ibu Fatah, Yusfir Rumi

Assistant Professor,CSE Dept., ahangirnagar University Savar, Bangladesh

Article Information

Received : 02 Sept 2024
Revised : 06 Sept 2024
Accepted : 18 Sept 2024
Published : 22 Sept 2024

Corresponding Author:
Jushak Prodhan

Abstract— Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication. Most cryptographic algorithms function more efficiently when implemented in hardware than in software running on single processor. However, systems that use hardware implementations have significant drawbacks: they are unable to respond to flaws discovered in the implemented algorithm or to changes in standards. As an alternative, it is possible to implement cryptographic algorithms in software running on multiple processors. However, 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.

Keywords: ***Cryptography, DES, AES, NIST, Plaintext, RSA.***

Citation: Jushak Prodhan, Ibu Fatah, Yusfir Rumi, "Performance Analysis of AES Techniques of Serial Implementation Algorithms", Journal of Science, Computing and Engineering Research, 7(9), Sep 2024.

I. INTRODUCTION

Cryptography is generally understood to be the study of the principles and techniques by which information is converted into an encrypted version that is difficult (ideally impossible) for any unauthorized person to convert to the original information, while still allowing the intended reader to do so. In fact, cryptography covers rather more than merely encryption and decryption. It is, in practice, a specialized branch of information theory with substantial additions from other branches of mathematics. Cryptography is probably the most important aspect of communications security [1] and is becoming increasingly important as a basic building block for computer security. The increased use of computer and communications systems by the industry has increased the risk of theft of proprietary information. Although these threats may require a variety of countermeasures, cryptography is a primary method of protecting valuable electronic information. In data and telecommunications, cryptography is necessary when communicating over any unsecured medium, which includes just about any network, particularly the Internet. Within the context of any application-to-application communication, there are some specific security requirements, including the following: Authentication: The process of proving one's identity. Confidentiality: Ensuring that no one can read the message except the intended receiver. Integrity: Assuring the receiver that the received message has not been altered in any way from the original. Non-repudiation: A mechanism to prove that the sender really sent this message. There are, in general, two types of cryptographic schemes typically used to accomplish these goals: secret key (or symmetric or conventional) cryptography and public-key (or asymmetric) cryptography. In symmetric-key cryptography, an algorithm is used to scramble the message using a secret key in such a way that it becomes unusable to all except the ones that have access to that secret key. The most widely known symmetric cryptographic algorithm is DES, developed by IBM in the seventies. It uses a key of 56 bits and operates on chunks of 64 bits at a time. In public key cryptography [4], algorithms use two different keys: a private and a public one. A message encrypted with a private key can be decrypted with its public key (and vice versa). The owner of the key pair holds the private key, and may distribute the public key to anyone. Someone who wants to send a secret message uses the public key of the intended receiver to encrypt it. Only the receiver holds the private key and can decrypt it.

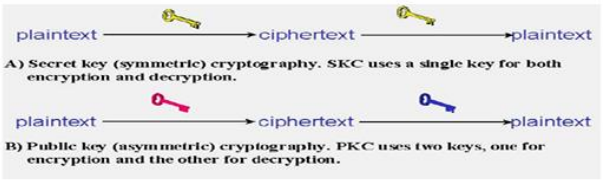


Figure 1: Two types of cryptography

II. DATA ENCRYPTION STANDARD (DES)

The most common symmetric-key cryptography scheme used today is the Data Encryption Standard (DES) [2], designed by IBM in the 1970s and adopted by the National Bureau of Standards (NBS) [now the National Institute for Standards and Technology (NIST)] in 1977 [2] for commercial and unclassified government applications. DES has been adopted as Federal Information Processing Standard 46 (FIPS 46-3) and by the American National Standards Institute as X3.92.

DES is a block-cipher employing a 56-bit key that operates on 64-bit blocks. DES has a complex set of rules and transformations that were designed specifically to yield fast hardware implementations and slow software implementations, although this latter point is becoming less significant today since the speed of computer processors is several orders of magnitude faster today than twenty years ago. IBM also proposed a 112-bit key for DES, which was rejected at the time by the government; the use of 112-bit keys was considered in the 1990s, however, conversion was never seriously considered.

1.2 AES: An Alternative to DES The symmetric-key cryptography is efficient for encryption while the Public-key cryptography facilitates efficient signatures (particularly non-repudiation) and key management. Symmetric key cryptography is faster than any currently available public-key encryption method. On the other hand, the most widely used symmetric-key encryption technique like DES is vulnerable to a brute-force attack [3] because of its inadequate key size compare to the processing power of modern computer.

In order to increase the security of symmetric-key cryptography, NIST in 1997 issued a call for proposals for a new Advanced Encryption Standard (AES), which should have security strength better than DES and significantly improved efficiency. In addition, to these general requirements, NIST specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits.

III. MATERIALS AND METHODS

AES Cipher: The Rijndael proposal for AES [6] 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.

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)	44/176	52/208	60/240

Table 1: AES Parameter

IV. OVERALL STRUCTURE OF AES

The overall structure of AES is depicted in figure 2. The input to the encryption and decryption algorithms is a single 128-bit block. 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: 3. 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. 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.

Several features of the overall AES structure [7]:

1. One noteworthy feature of this structure is that it is not a Feistel structure. In the classic feistel structure, half of the data block is used to modify the other half of the data block, and then the half are swapped. Rijndael does not use a Feistel structure but process the entire block in parallel during each round using substitutions and permutations.

2. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round; these are indicated in Figure 2. 3. Four different stages are used, one of permutation and three of substitution: Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block. Shift rows: A simple permutation Mix Columns: A substitution that makes use of arithmetic over GF (28) Add round Key: A simple bitwise XOR of the current block with a portion of the expanded

key

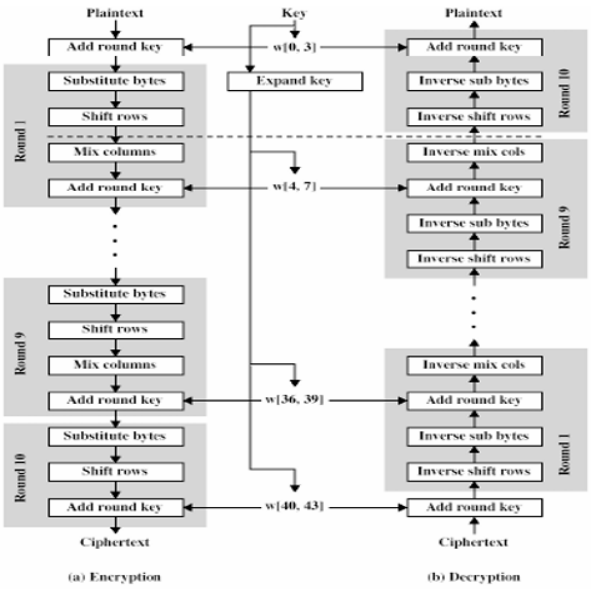
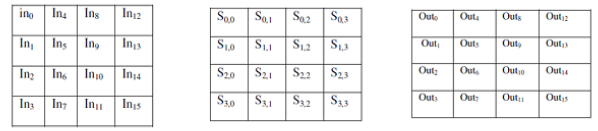


Figure 2: AES encryption and decryption

4. The structure of AES is quite simple. For both encryption and decryption, the cipher begins with an Add Round Key stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages. Figure 4 depicts the structure of a full encryption round. 5. Only the Add Round Key stage makes use of the key. For this reason, the cipher begins and ends with an Add Round Key stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.



(a) Input, State array and Output

(b) Key and Expanded Key

Figure 3: AES Data Structure

6. The Add Round Key stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. The cipher is an alternating operations of XOR encryption (Add Round Key) of a block, followed by scrambling of the block (the other three stages), and followed by XOR encryption, and so on. This scheme is both efficient and highly secure. 7. Each stage is easily reversible. For the Substitute Byte, Shift Row, and Mix Columns stages, an inverse function is used in the decryption algorithm. For the Add Round Key stage, the inverse is achieved by XORing the same round key to the block, using the result that A ⊕ A ⊕ B = B. 8. As with most block ciphers [5], the decryption algorithm makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm. This is a consequence of the particular structure of AES. 9. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext. Figure 2 lays out encryption and decryption going in opposite vertical directions. At each horizontal point (e.g.,the dashed line in the figure), State is the same for both encryption and decryption. 10. The final round of both encryption and decryption consists of only three stages. Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible.

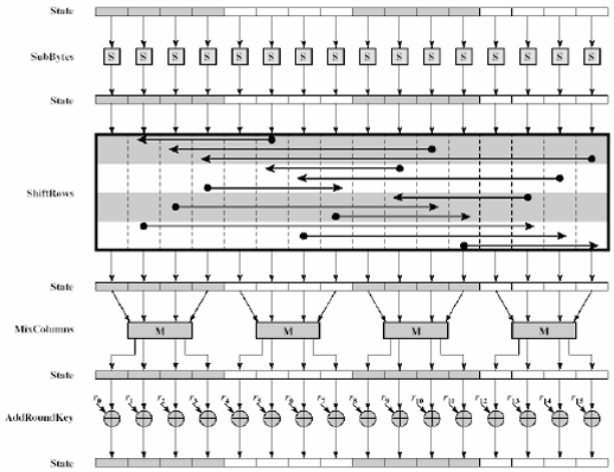


Figure 4: AES encryption round

V. ALGORITHM FOR SERIAL IMPLEMENTATION OF AES

AES is an iterated block cipher, meaning that the initial input block and cipher key undergoes multiple rounds of transformation before producing the output. Each intermediate cipher result is called a State. The block and cipher key are often represented as an array of columns where each array has 4 rows and each column represents a single byte (8 bits). The number of columns in an array representing the state or cipher key, then, can be calculated as the block or key length divided by 32 (32 bits = 4 bytes). An array representing a State will have Nb columns, where Nb values of 4, 6, and 8 correspond to a 128-, 192-, and 256-bit block, respectively. Similarly, an array representing a Cipher Key will have Nk columns, where Nk values of 4, 6, and 8 correspond to a 128-, 192-, and 256-bit key,

respectively. The AES cipher itself has three operational stages: 1. AddRound Key transformation 2. Nr-1 Rounds comprising: • SubBytes transformation • ShiftRows transformation • MixColumns transformation • AddRoundKey transformation 3. A final Round comprising: • SubBytes transformation • ShiftRows transformation • AddRoundKey transformation The overall structure of AES cipher is described below: Constants: int Nb = 4; int Nr = 10, 12, or 14; // rounds, for Nk = 4, 6, or 8 Inputs: array in of 4*Nb bytes // input plaintext array out of 4*Nb bytes // output ciphertext array w of 4*Nb*(Nr+1) bytes // expanded key Internal work array: state, 2-dim array of 4*Nb bytes, 4 rows and Nb cols

Algorithm :

```
void Cipher(byte[] in, byte[] out, byte[] w) {
byte[][] state = new byte[4][Nb];
state = in;
AddRoundKey(state, w, 0, Nb - 1);
for (int round = 1; round < Nr; round++) {
SubBytes(state);
ShiftRows(state);
MixColumns(state);
AddRoundKey(state, w, round*Nb, (round+1)*Nb - 1);
} SubBytes(state);
ShiftRows(state);
AddRoundKey(state, w, Nr*Nb, (Nr+1)*Nb - 1);
out = state; }
```

In the above algorithm:

- *in[]* and *out[]* are 16-byte arrays with the plaintext and cipher text, respectively. Both of these arrays are actually 4***Nb** bytes in length but **Nb**=4 in AES.
- *state[]* is a 2-dimensional array containing bytes in 4 rows and 4 columns. This arrays is 4 rows by **Nb** columns.
- *w[]* is an array containing the key material and is 4*(**Nr**+1) words in length.
- *AddRoundKey()*, *SubBytes()*, *ShiftRows()*, and *MixColumns()* are functions representing the individual transformations.

2.3 Run Time Complexity of the Serial Implementation

The number of steps an algorithm requires to solve a specific problem is denoted as the running time of the algorithm. In general, the running time depends on the size of the problem and on the respective input. In order to evaluate an algorithm independently of the input, the notation of time complexity is introduced. The time complexity T(n) is a function of the problem size n. The value of T(n) is the running time of the algorithm in the worst case, i.e. the number of steps it requires at most with an arbitrary input. However, time complexity function does not give the actual execution time of an algorithm rather it gives an idea how the time required for an algorithm changes as the problem size increases.

In order to compute the run time complexity of the AES algorithm, the time complexity function for each transformation has to be considered. As the AES algorithm consists of only four different types of transformation, the time complexity function of AES will depend on the time complexity of each transformation. From the time complexity of different transformation function, it is found that the AES algorithm has a linear complexity that means when the value of N (number of data block) ranges from 10 – 100000, the execution time will vary from 10-5 second to 1 seconds (each operation is assumed to take 10-6 second). However, when the value of N is greater than 108, the execution time of the algorithm will require several days to encrypt or decrypt. The following table will give a clear idea:

2.4 Computer time used for different data blocks

Problem Size (No. of data blocks)	No. of Operations	Time Required
10	10	10^{-5} Sec
10^2	100	10^{-4} Sec
10^3	1000	10^{-3} Sec
10^4	10000	10^{-2} Sec
10^5	100000	10^{-1} Sec
10^6	1000000	1 Sec
10^{10}	10000000000	2.78 hours
10^{12}	1000000000000	11.57 days

2.5 PARALLEL IMPLEMENTAION OF AES

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. 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. For implementing the AES algorithm in parallel, the MPI based cluster is used in the present section. The performance of a parallel algorithm depends not only on input size but also on the architecture of the parallel computer, the number of processors, and the interconnection network. In this chapter, different types of parallel architectures and interconnection networks are discussed before actually implementing the parallel algorithm of AES. At the end of this chapter, some sample input/output are shown varying the key size, number of rounds and the number of processors to verify the correctness of parallel algorithm. Finally, the run time complexity of the parallel algorithm is shown to measure the performance improvement of the parallel implementation over the serial implementation.

2.6 Algorithm for Parallel Implementation of AES

There are two major components of parallel algorithm design. 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. 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. Ideally, a parallel algorithm should have maximum concurrency and locality. However, for most algorithms, there is a tradeoff. An algorithm that has more concurrency often has less locality. To implement the AES algorithm in parallel, data blocks (Figure 5) 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. After encrypting all the data blocks of each processor, the encrypted data will be merged 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.

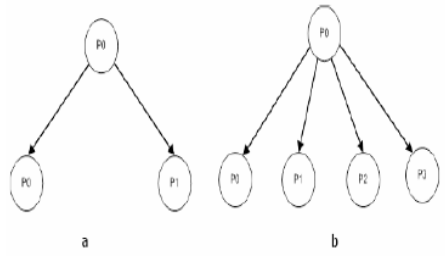


Figure 5: (a, b) Data blocks are distributed between two and four processors

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. The overall parallel algorithm of AES cipher is described below: Constant: ArraySize = 160 ; int Nb = 4; int Nr = 10, 12, or 14; // rounds, for Nk = 4, 6, or 8

Inputs: int nProcessors = 2/4/8/16 processors int tNumberOfBlocks // number of blocks to be encrypted unsigned char key[16] // key for encrypting data int k = 0; array w of 4*Nb*(Nr+1) bytes // expanded key Internal work array: my_pointer is an array of pointers where each element of the array points to an array of data blocks. Each processor will have the variable my_pointer, where the first index will contain the data blocks for each processor. Algorithm: void Cipher(byte[] in, byte[] out, byte[] w) { nProcessors = 4 int nBlockPerProcessor = tNumberOfBlocks / nProcessors int rank = processor's label if (rank = = processor 0){ my_pointer[0] = nBlockPerProcessor data blocks read the key send nBlockPerProcessor data blocks to rest of the processors send the same key to other processors } else{// for all other processors receive the nBlockPerProcessor data blocks from processor 0 my_pointer[0] = nBlockPerProcessor data blocks receive the key from the processor 0 } // each processor will execute this part of the algorithm. //Encrytion Encryption(my_pointer[0]); // Encrypted data are merged in tree structure and return back to the // main processor BTM(0, nProcessors -1); //Decryption Decryption(my_pointer[1]); BTM(0, nProcessors -1);}

3. SAMPLE INPUT/OUTPUT 128-bit data, 128-bit Key 2 processors, each processor processes 4 data blocks Encrypting . . .

Processor 0:

Plaintext1	32	43	F6	A8	88	5A	30	8D	31	31	98	A2	E0	37	07	34
Plaintext2	38	21	1A	00	0B	23	DE	93	F7	B6	65	7D	F9	AE	C4	D1
Plaintext3	AF	DA	94	A5	E5	3C	A1	25	B0	39	D3	58	0	CE	BF	CA
Plaintext4	8E	9C	32	1E	84	47	CD	BC	9B	67	7E	B9	B6	23	5E	1A
Key	2B	7E	15	16	28	AE	D2	A6	AB	F7	15	88	09	CF	4F	3C
Ciphertext1	39	25	84	1D	02	DC	09	FB	DC	11	85	97	19	6A	0B	32
Ciphertext2	E3	97	D6	93	C8	82	E9	DF	8A	CD	3D	35	2A	20	0A	47
Ciphertext3	71	FB	5E	D7	3E	D0	76	AE	C5	A1	89	14	86	70	16	3F
Ciphertext4	DB	CA	D3	9F	C2	FC	B6	EF	F5	1B	60	39	53	1B	2B	24

Processor 1:

Plaintext1	D1	5B	5F	58	91	EA	82	C0	1F	28	4B	1A	37	B3	73	89
Plaintext2	3F	91	38	5A	E1	F3	7B	9C	3D	C2	AA	7B	9B	F2	7C	23
Plaintext3	7C	70	DC	B2	0F	CC	CA	20	5F	48	4F	E7	43	34	07	88
Plaintext4	46	BA	06	15	41	1F	0F	96	30	46	06	AA	3E	76	A8	72
Key	2B	7E	15	16	28	AE	D2	A6	AB	F7	15	88	09	CF	4F	3C
Ciphertext1	B4	FD	4B	E9	64	FF	4F	80	84	59	24	88	0F	21	54	F5
Ciphertext2	23	20	B7	96	CC	27	7A	91	E4	CC	1D	48	D4	75	3C	44
Ciphertext3	BC	BD	C4	72	EE	F1	A7	9F	51	FF	C3	2A	E7	B1	52	7C
Ciphertext4	0F	E9	FB	87	42	0F	AA	DD	0C	C6	9C	E1	40	F5	8B	E4

Decrypting . . .
Processor 0:

Ciphertext1	39	25	84	1D	02	DC	09	FB	DC	11	85	97	19	6A	0B	32
Ciphertext2	E3	97	D6	93	C8	82	E9	DF	8A	CD	3D	35	2A	20	0A	47
Ciphertext3	71	FB	5E	D7	3E	D0	76	AE	C5	A1	89	14	86	70	16	3F
Ciphertext4	DB	CA	D3	9F	C2	FC	B6	EF	F5	1B	60	39	53	1B	2B	24
Key	2B	7E	15	16	28	AE	D2	A6	AB	F7	15	88	09	CF	4F	3C

Plaintext1	32	43	F6	A8	88	5A	30	8D	31	31	98	A2	E0	37	07	34
Plaintext2	38	21	1A	00	0B	23	DE	93	F7	B6	65	7D	F9	AE	C4	D1
Plaintext3	AF	DA	94	A5	E5	3C	A1	25	B0	39	D3	58	00	CE	BF	CA
Plaintext4	8E	9C	32	1E	84	47	CD	BC	9B	67	7E	B9	B6	23	5E	1A
Processor 1:																
Ciphertext1	B4	FD	4B	E9	64	FF	4F	80	84	59	24	88	0F	21	54	F5
Ciphertext2	23	20	B7	96	CC	27	7A	91	E4	CC	1D	48	D4	75	3C	44
Ciphertext3	BC	BD	C4	72	EE	F1	A7	9F	51	FF	C3	2A	E7	B1	52	7C
Ciphertext4	0F	E9	FB	87	42	0F	AA	DD	0C	C6	9C	E1	40	F5	8B	E4
Key	2B	7E	15	16	28	AE	D2	A6	AB	F7	15	88	09	CF	4F	3C
Plaintext1	D1	5B	5F	58	91	EA	82	C0	1F	28	4B	1A	37	B3	73	89
Plaintext2	3F	91	38	5A	E1	F3	7B	9C	3D	C2	AA	7B	9B	F2	7C	23
Plaintext3	7C	70	DC	B2	F	CC	CA	20	5F	48	4F	E7	43	34	7	88
Plaintext4	46	BA	6	15	41	1F	F	96	30	46	6	AA	3E	76	A8	72

VI. CONCLUSION

In the figure 6 and 7, the performance of serial and parallel implementation of AES is shown with 2 processors. The speedup factor of AES is given in figure 8 with 2 processors.

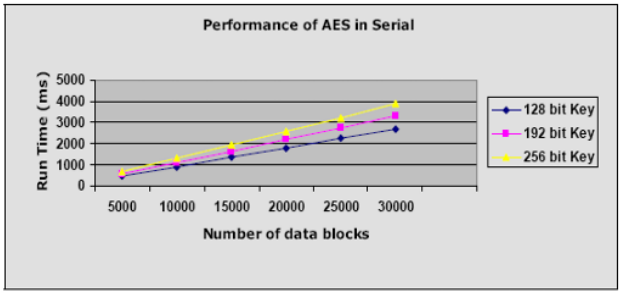


Figure 6: Performance of AES in Serial

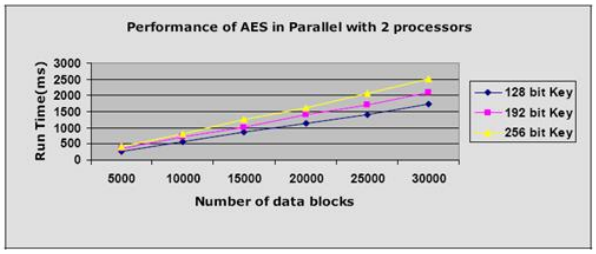


Figure 7: Performance of AES in Parallel with 2 processors

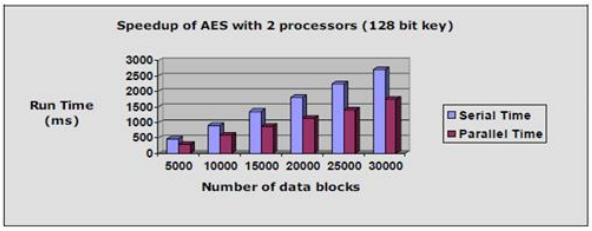


Figure 8: Speedup of AES with 2 processors

After implementing the AES algorithm on a single processor, it is found that the AES algorithm has a linear complexity that means when the value of N (number of data blocks) ranges from 10 – 100000, the execution time will vary from 10-5 second to 1 seconds (each operation is assumed to take 10-6 second). However, when the value of N is greater than 108, the execution time of the algorithm will require several days to encrypt or decrypt. This creates the reason for implementing the algorithm in parallel. After implementing the AES algorithm in parallel, it is found that the performance of AES algorithm increases significantly as the number of processor increases. It is not possible to get the speedup factor equal to P (number of processor), as some parallel processing overhead is also occurred during the implementation of AES in parallel.

REFERENCES

[1]. P. Nirmala, T. Manimegalai, J. R. Arunkumar, S. Vimala, G. Vinoth Rajkumar, Raja Raju, "A Mechanism for Detecting the Intruder in the Network through a Stacking Dilated CNN Model", Wireless Communications and Mobile Computing, vol. 2022, Article ID 1955009, 13 pages, 2022. https://doi.org/10.1155/2022/1955009.

[2]. D. Sathyanarayanan, T. S. Reddy, A. Sathish, P. Geetha, J. R. Arunkumar and S. P. K. Deepak, "American Sign Language Recognition System for Numerical and Alphabets," 2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE), Chennai, India, 2023, pp. 1-6, doi: 10.1109/RMKMATE59243.2023.10369455.

[3]. J. R. Arunkumar, Tagele berihun Mengist, 2020" Developing Ethiopian Yirgacheffe Coffee Grading Model using a Deep Learning Classifier" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-9 Issue-4, February 2020. DOI: 10.35940/ijitee.D1823.029420.

[4]. Ashwini, S., Arunkumar, J.R., Prabu, R.T. et al. Diagnosis and multi-classification of lung diseases in CXR images using optimized deep convolutional neural network. Soft Comput (2023). https://doi.org/10.1007/s00500-023-09480-3

[5]. J.R.Arunkumar, Dr.E.Muthukumar," A Novel Method to Improve AODV Protocol for WSN" in Journal of Engineering Sciences" ISSN NO: 0377-9254Volume 3, Issue 1, Jul 2012.

[6]. R. K, A. Shameem, P. Biswas, B. T. Geetha, J. R. Arunkumar and P. K. Lakineni, "Supply Chain Management Using Blockchain: Opportunities, Challenges, and Future Directions," 2023 Second International Conference on Informatics (ICI), Noida, India, 2023, pp. 1-6, doi: 10.1109/ICI60088.2023.10421633.

[7]. Arunkumar, J. R. "Study Analysis of Cloud Security Chanllenges and Issues in Cloud Computing Technologies." Journal of Science, Computing and Engineering Research 6.8 (2023): 06-10.

[8]. J. R. Arunkumar, R. Raman, S. Sivakumar and R. Pavithra, "Wearable Devices for Patient Monitoring System using IoT," 2023 8th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2023, pp. 381-385, doi: 10.1109/ICCES57224.2023.10192741.

[9]. S. Sugumaran, C. Geetha, S. S, P. C. Bharath Kumar, T. D. Subha and J. R. Arunkumar, "Energy Efficient Routing Algorithm with Mobile Sink Assistance in Wireless Sensor Networks," 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2023, pp. 1-7, doi: 10.1109/ACCAI58221.2023.10201142.

[10]. R. S. Vignesh, V. Chinnammal, Gururaj.D, A. K. Kumar, K. V. Karthikeyan and J. R. Arunkumar, "Secured Data Access and Control Abilities Management over Cloud Environment using Novel Cryptographic Principles," 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2023, pp. 1-8, doi: 10.1109/ACCAI58221.2023.10199616.

[11]. Syamala, M., Anusuya, R., Sonkar, S.K. et al. Big data analytics for dynamic network slicing in 5G and beyond with dynamic user preferences. Opt Quant Electron 56, 61 (2024). https://doi.org/10.1007/s11082-023-05663-2

[12]. Krishna Veni, S. R., and R. Anusuya. "Design and Study Analysis Automated Recognition system of Fake Currency Notes." Journal of Science, Computing and Engineering Research 6.6 (2023): 16-20.

[13]. V. RamKumar, S. Shanthi, K. S. Kumar, S. Kanageswari, S. Mahalakshmi and R. Anusuya, "Internet of Things Assisted Remote Health and Safety Monitoring Scheme Using Intelligent Sensors," 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2023, pp. 1-8, doi: 10.1109/ACCAI58221.2023.10199766.

[14]. R. S. Vignesh, R. Sankar, A. Balaji, K. S. Kumar, V. Sharmila Bhargavi and R. Anusuya, "IoT Assisted Drunk and Drive People Identification to Avoid Accidents and Ensure Road Safety Measures," 2023 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI), Chennai, India, 2023, pp. 1-7, doi: 10.1109/ACCAI58221.2023.10200809.

[15]. I. Chandra, G. Sowmiya, G. Charulatha, S. D, S. Gomathi and R. Anusuya, "An efficient Intelligent Systems for Low-Power Consumption Zigbee-Based Wearable Device for Voice Data Transmission," 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India, 2023, pp. 1-7, doi: 10.1109/ICECONF57129.2023.10083856.

[16]. G. Karthikeyan, D. T. G, R. Anusuya, K. K. G, J. T and R. T. Prabu, "Real-Time Sidewalk Crack Identification and Classification based on Convolutional Neural Network using Thermal Images," 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 2022, pp. 1266-1274, doi: 10.1109/ICACRS55517.2022.10029202.

[17]. R. Meena, T. Kavitha, A. K. S, D. M. Mathew, R. Anusuya and G. Karthik, "Extracting Behavioral Characteristics of College Students Using Data Mining on Big Data," 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India, 2023, pp. 1-7, doi: 10.1109/ICECONF57129.2023.10084276.

[18]. S. Bharathi, A. Balaji, D. Irene. J, C. Kalaivanan and R. Anusuya, "An Efficient Liver Disease Prediction based on Deep Convolutional Neural Network using Biopsy Images," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022, pp. 1141-1147, doi: 10.1109/ICOSEC54921.2022.9951870.

[19]. I. Chandra, G. Sowmiya, G. Charulatha, S. D, S. Gomathi and R. Anusuya, "An efficient Intelligent Systems for Low-Power Consumption Zigbee-Based Wearable Device for Voice Data Transmission," 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India, 2023, pp. 1-7, doi: 10.1109/ICECONF57129.2023.10083856.

[20]. Revathi, S., et al. "Developing an Infant Monitoring System using IoT (INMOS)." International Scientific Journal of Contemporary Research in Engineering Science and Management 6.1 (2021): 111-115.

[21]. J.R.Arunkumar, Dr.E.Muthukumar,‖ A Novel Method to Improve AODV Protocol for WSN‖ in Journal of Engineering Sciences‖ ISSN NO: 0377-9254Volume 3, Issue 1, Jul 2012.

[22]. R. S. Vignesh, A. Kumar S, T. M. Amirthalakshmi, P. Delphy, J. R. Arunkumar and S. Kamatchi, "An Efficient and Intelligent Systems for Internet of Things Based Health Observance System for Covid 19 Patients," 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India, 2023, pp. 1-8, doi: 10.1109/ICECONF57129.2023.10084066.

[23]. I. Chandra, K. V. Karthikeyan, R. V, S. K, M. Tamilselvi and J. R. Arunkumar, "A Robust and Efficient Computational Offloading and Task Scheduling Model in Mobile Cloud Computing," 2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF), Chennai, India, 2023, pp. 1-8, doi: 10.1109/ICECONF57129.2023.10084293.

[24]. R. K, A. Shameem, P. Biswas, B. T. Geetha, J. R. Arunkumar and P. K. Lakineni, "Supply Chain Management Using Blockchain: Opportunities, Challenges, and Future Directions," 2023 Second International Conference on Informatics (ICI), Noida, India, 2023, pp. 1-6, doi: 10.1109/ICI60088.2023.10421633.

[25]. J. R. Arunkumar, and R. Anusuya, “OCHRE: A Methodology for the Deployment of Sensor Networks.” American Journal of Computing Research Repository, vol. 3, no. 1 (2015): 5-8.