VFAST Transactions on Mathematics
http://vfast.org/journals/index.php/VTM@ 2020 ISSN(e): 2309-0022; ISSN(p): 2411-6343
Volume 8, Number 1, January-December, 2020
pp. 30-37

PARALLELIZATION OF ENCRYPTION ALGORITHMS USING MPI

Dr. Muhammad Hanif Durad[1], Ahmad Raza[2], Ali Asad [3], Muhammad Naveed Akhtar[4], Department of Computer and Information Science (DCIS),
Pakistan Institute of Engineering & Applied Sciences (PIEAS)
[1]hanif@pieas.edu.pk, [2]mphilcs1302@ pieas.edu.pk , [3] mphilcs1303@ pieas.edu.pk, [4]naveed@ pieas.edu.pk

Abstract—Encryption is the basic technique to achieve data confidentiality. A number of algorithms including RSA, DES, Blowfish and AES have been parallelized using MPI and have been employed for practical file encryption. A unified performance analysis of all these algorithms has been presented using two ECB and counter modes. On basis of experimental results some guidelines has been suggested for the end user to select appropriate algorithm for achieving enhanced speed up.
Keywords— Data Encryption Standard (DES), Advanced Encryption Standard (AES), Electronic Codebook (ECB) mode, Counter (CTR) mode, Parallel encryption.

1. **Introduction**. Encryption is means to transform plain text into cipher text, and designing parallel algorithms for encryption may be necessary in near future with the advent of multi-core processors. Parallel processing may be used to enhance the speedup for the file encryption for security purpose for transmission and reception of confidential data. In MPI the process are independent of each other, they can be executed in parallel, and hence they can be applied in parallel to the different parts of the file.
In this paper both symmetric and asymmetric algorithms [1], have been implemented for parallel file encryption. The asymmetric algorithm is RSA where as DES, Blowfish and AES. The input file is divided into slices of data as permitted by the size of cipher input and each data chunk is executed by a single process. All algorithms have been implemented in Electronic Codebook (ECB) and Counter (CTR) modes except RSA which was used in ECB mode due to private public key concept involved. No security enhancements have been suggested in this paper for certain encryption modes. The objective is to get the idea of speed enhancement for practical user's point of view. In this paper we don't focus on internal parallelization of algorithm like [2] which uses OpenMP.

The rest of this paper is structured as follows: Section II summarizes related work, in section III describes the target architectures and execution environment used for performance evaluation, while section IV reviews the individual parallel encryption algorithms. Section IV presents unified analysis all algorithms based on experimental results. Section V concludes the paper.

2. **Related Work**. Parallel cryptography and cryptanalysis is comparatively new field in computer science domain. A lot of efforts needs to be put forward especially in the era of cloud computing. Internal parallelization of cryptosystems is also very crucial due to modern multi-core processors; a few efforts are described in following paragraphs.

Beletskyy, V., and D. Burak [2] DES using OpenMP. They have concentrated on parallelization of internal loops in DES, they have achieved a speed-up is about 1.95 just using two processors machine.

Niederhagen, Ruben, and Jen-Hsun Huang [3] have Graphics Processing Units (GPUs) for Parallel Cryptanalysis. They have investigated three different attacks on particular cryptographic systems including generalized birthday attack is applied to the compression function of the Fast Syndrome-Based (FSB) hash function, Pollard's rho algorithm used for attacking Certicom's ECC Challenge ECC2K-130 and XL algorithm.

Das, Debasis, and Abhishek Ray [4] have suggested use cellular automata (CA) in encryption which are highly parallel and discrete dynamical systems. They have used CA in cryptography for a class of block ciphers through a suggesting a new block encryption algorithm.

Hashem Mohammed Alaidaros et. al. [5] has employed Hashed Message Authentication Code (HMAC) and AES using MPI in the Secure Socket Layer (SSL) implementation. Karthikeyan, S. et. al [6] have implemented AES in parallel using MPI similar to this paper but scope of their article is much smaller than this writing.

In summary, the existing papers have following limitations:

- It is difficult to get a unified picture of the performance various parallel encryption algorithms using MPI.
- Only a few encryption algorithms have been implemented using MPI.
- Analysis of comparatively small sized file is present in literature [6] this can't serve as guideline for proper selection of a particular algorithm for end user.

In the nutshell, we believe that the paper will provide some extensions in theoretical background and practical implementation of parallel encryption algorithms.

3. **Target Architectures And Execution Environment**. The machine namely a computing cluster have used in experiments having the following architectures:

Computing Cluster:
Head Node: 2 x Intel Xeon Processors E5504 @ 2.00 GHz with 4 MB cache, 4 cores and 16 GB memory,
Cluster-Workers: 6 x Intel Core i5 Processors @ 2.67GHz with 8MB Cache, 4 cores and 4 GB memory each.
These algorithms were executed repeatedly on above system using MPI. The computing cluster system has 40 processing elements out of which 8 utilized in the computational experiments carried in this paper. These processing elements communicate with each other over Gigabit Ethernet network. Input files used were created using uniform random number generator then converted in text format for intelligibility and decryption. Their sizes were varied from 256KB to 64 MB. During execution processing elements were grouped as 1, 2, 4, and 8.

4. Review and performance of individual parallel sorting algorithms. A large number of encryption algorithms are available in literature. But only a few algorithms have been parallelized. The paper considers both symmetric and asymmetric algorithms. For computational analysis in this section different file sizes have been used for said cluster system.

A. ***RSA***

RSA One of the first public-key schemes was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978 [7]. The RSA scheme is considered to be the most widely accepted and implemented approach to public-key encryption. RSA is a block cipher where data is mapped to numbers. The interested reader can refer [1]**.** for more discussion. RSA has been used in ECB mode, since CTR mode can't be implemented due to the concept of private and a public key involved in this cipher.

The following Fig. 1 shows the execution time for said data sets using RSA in counter mode.
Fig. 1 depicts the usual MPI trend as the number of processing elements increases the execution time for RSA file encryption decreases. But it can be seen from Fig. 1 for small inputs (512KB-64MB), the execution time for a relatively small file may be larger than a large input file.

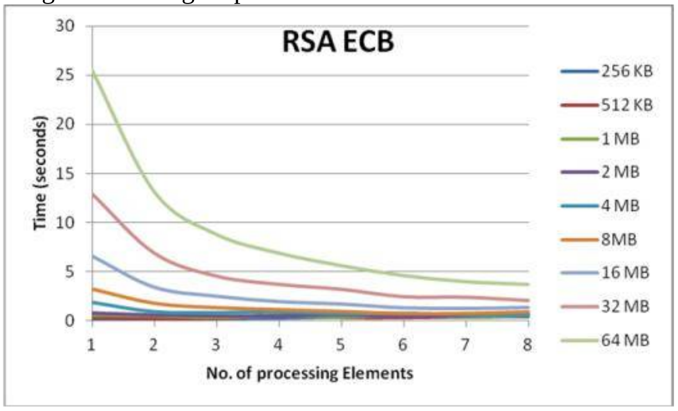

Fig. 1 Execution time for RSA in CTR mode

We believe this is due to asymmetric nature of RSA and the exponentiation process involved. Another notable feature of RSA is it is very sluggish in performing encryption and can't be used for file encryption .

B. ***DES***

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46).) [1]..

Fig. 2 shows the execution time for DES ECB mode using same files.

It can be seen from Fig. 2 due to symmetric nature of DES as the file size increases the execution time increase for the same number of processing elements. However for the same file size the execution time decrease with increase in processing elements.

Fig. 3 shows the execution time for DES CTR mode using

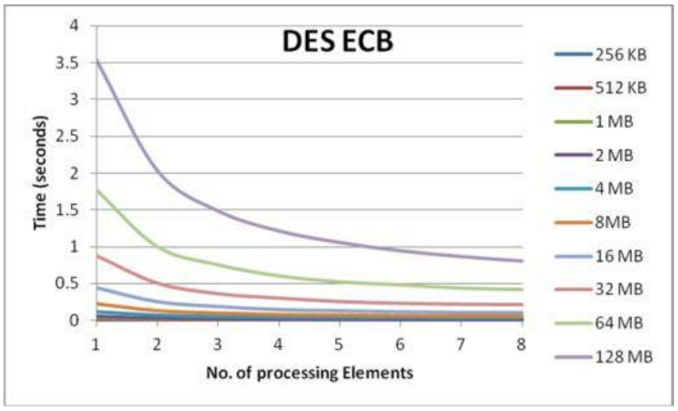

Fig. 2 Execution time for DES in ECB mode

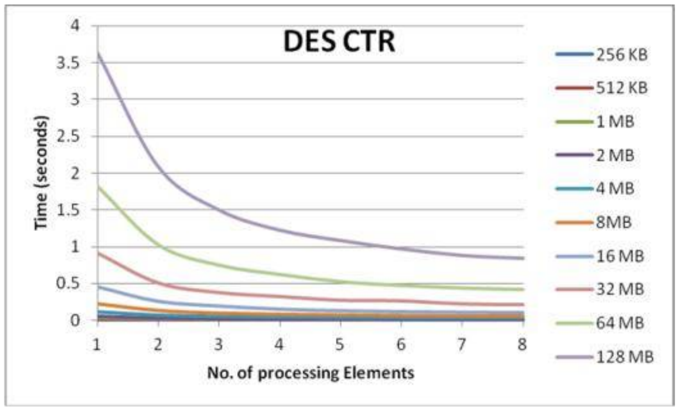

Fig. 3 Execution time for DES in CTR mode

It can be seen from Fig. 3 the same trend continues as in Fig. 2 Execution time for DES in ECB mode.

C. ***AES***

The Advanced Encryption Standard (AES) was issued as a federal information processing standard (FIPS 197). It is proposed to replace DES and triple DES with an algorithm that is more secure and efficient. AES uses a block length of 128 bits and a key length that can be 128, 192, or 256 bits. In the description of this section, we assume a key length of 128 bits, which is likely to be the one most commonly implemented. The interested reader can refer [1]**,** for more discussion. Fig. **4** shows the execution time for AES ECB mode using same files.

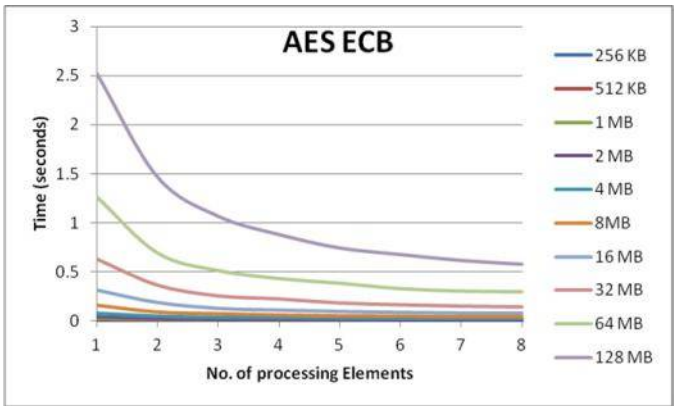

Fig. 4 Execution time for AES in ECB mode

Fig. 5 shows the execution time for AES CTR mode using same files.

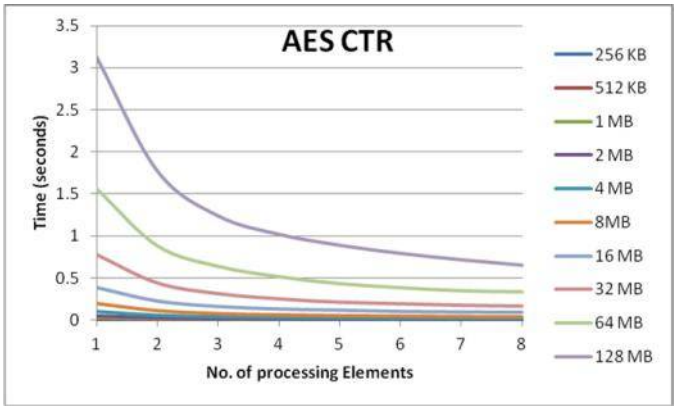

Fig. 5 Execution time for AES in CTR mode

It can be seen from Fig. **4** and Fig. 5 the same trend continues as for DES.

D. ***Blowfish***

Blowfish, a secret-key block cipher, proposed in [9]. It uses Feistel structure, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Though there is a complex initialization phase needed before any encryption can occur, the actual encryption of data is very competent on large microprocessors. The interested reader can refer [1], for more discussion.

Fig. **6** shows the execution time for Blowfish in ECB mode using same files.

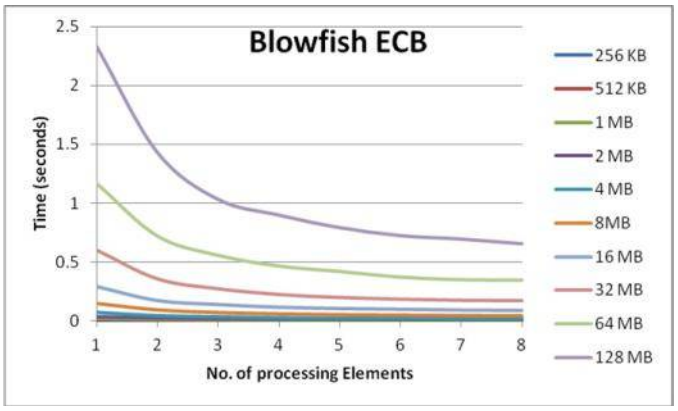

Fig. 6 Execution time for Blowfish in ECB mode

Fig. 7 shows the execution time for blowfish CTR mode using same files.

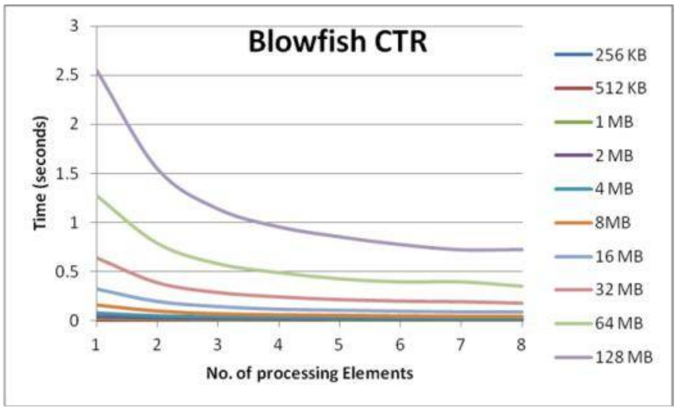

Fig. 7 Execution time for Blowfish in CTR mode

It can be seen from
Fig. **6** and Fig. 7 the same trend continues as DES and AES.

5. Unified Performance Analysis for cluster system. A unified performance analysis of various parameters for DES, AES and Blowfish algorithms has been performed and the results are presented in this section. Please note RSA has been excluded due to its bad performance for file encryption:

A. Execution time versus number of processing elements

A fixed file size of 128MB all said encryption algorithms were used for 1, 2, 4, and 8 processing elemets groups to observe execution time, the results are shown in Fig. 8 for ECB modes.

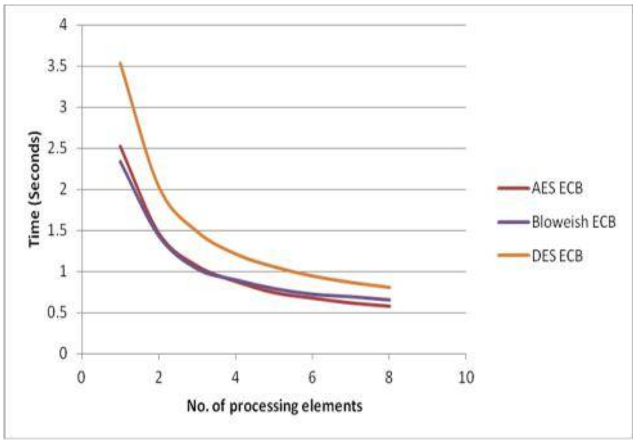

Fig. 8 Execution time vs no of processing elements for ECB mode

It can be seen from Fig. 8 the DES outperforms AES and Blowfish in ECB mode whereas Blowfish and AES has similar performance. We judge this is due to less computation involved in as compared AES and Blowfish.

For the same sized file as in Fig. 8, the timing results are shown in Fig. 9 for CTR mode.

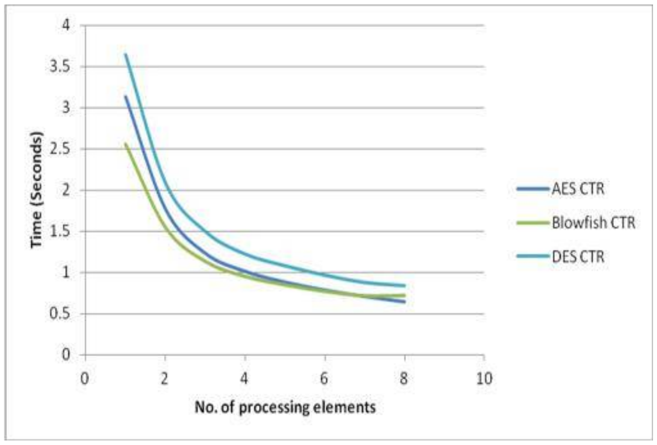

Fig. 9 Execution time vs no of processing elements for CTR mode

It can be seen again from Fig. 9, again similar performances are given by all algorithms as in Fig. 8.

B. ***Execution time versus File Size***

All said encryption algorithms have been analyzed by varying file size, the results are shown for ECB and CTR modes in Fig. 10 and Fig. 11.

It can be seen from Fig. 10 and Fig. 11 the DES perform better as compared Blowfish and AES both ECB and CTR modes as explained earlier.

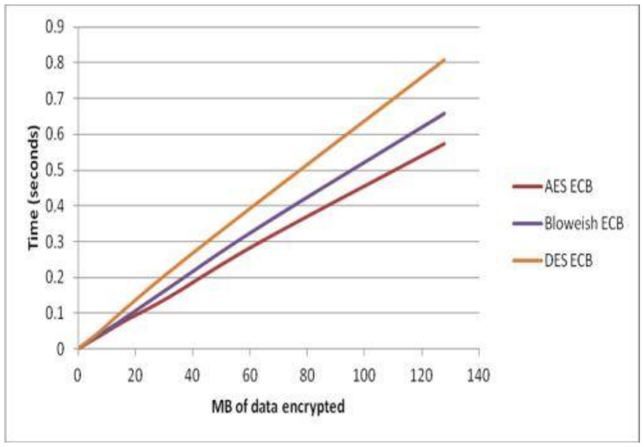

Fig. 10 Execution time vs. File Size (MB) for ECB mode

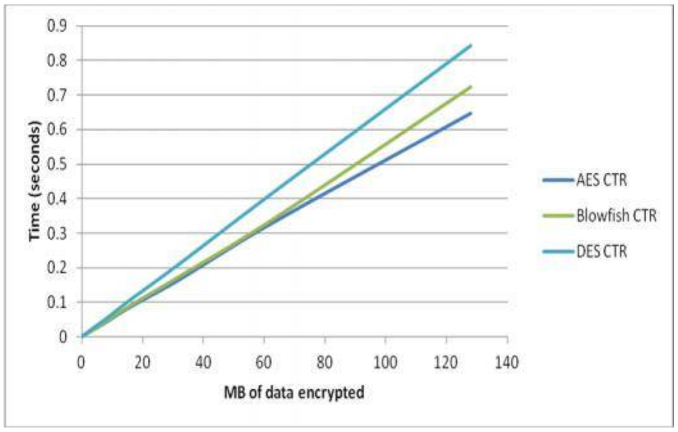

Fig. 11 Execution time vs. File Size (MB) for CTR mode

6. Summary. In this paper we have implemented and analyzed RSA, DES, AES and Blowfish encryption algorithms using MPI. It is very difficult to find a single paper implementing all these sorting methods altogether. Our findings from computational experiments performed in this regard are listed as:

1. There is a general decrease in execution time with increase in no of processing elements, encryption methods.
2. DES performs better than Blowfish and AES.
3. RSA is very sluggish due it asymmetric nature and can't be used for file encryption. We had tried to get overall picture of RSA, DES, AES and Blowfish encryption algorithms for parallelization, but we intent to extent our work by considering the following possible additions.

1. We are more interested extend ECB for security enhancement by adding some key scheduling algorithm for each file block. Of course this will require same MPI based algorithm to be used for decryption as well. Such built in hard coded processors may available in near future.
2. Security weakness of counter mode pointed in [9] may be overcome by using a parallel random number generator instead of simple increment operation in this mode.
3. Other parallel library standards such as OpenMP and Pthreads may be implemented due to impoertance multicore architecture.

In short, we consider that this paper has contributed to a few extensions in practical implementation of parallel encryption algorithms.

References

[1] Stallings, W. (2006). *Cryptography and network security, 4/E*. Pearson Education India.

[2] Beletskyy, V., & Burak, D. (2005). Parallelization of the data encryption standard (DES) algorithm. In *Enhanced methods in computer security, biometric and artificial intelligence systems* (pp. 23-33). Springer, Boston, MA.

[3] Niederhagen, R. F. (2012). Parallel cryptanalysis.

[4] Das, D., & Ray, A. (2010). A parallel encryption algorithm for block ciphers based on reversible programmable cellular automata. *arXiv preprint arXiv:1006.2822*.

[5] Alaidaros, H. M., Rasid, M. F. A., Othman, M., & Abdullah, R. S. A. R. (2007, May). Enhancing security performance with parallel crypto operations in ssl bulk data transfer phase. In *2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications* (pp. 129-133). IEEE.

[6] Manikandan, G., Sairam, N., & Kamarasan, M. (2012). A new approach for improving data security using iterative blowfish algorithm. *Research Journal of Applied Sciences, Engineering and Technology*, *4*(6), 603-607.

[7] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120-126.

[8] Amato, N. M., Iyer, R., Sundaresan, S., & Wu, Y. (1996). A comparison of parallel sorting algorithms on different architectures. *Technical Report TR98-029, Department of Computer Science, Texas A&M University*.

[9] Schneier, B. (1993, December). Description of a new variable-length key, 64-bit block cipher (Blowfish). In *International Workshop on Fast Software Encryption* (pp. 191-204). Springer, Berlin, Heidelberg.

[10] McGrew, D. A. (2002). Counter mode security: Analysis and recommendations. *Cisco Systems, November*, *2*(4).