International Journal of Computer Applications (0975 – 8887)
Volume 179 – No.50, June 2018

Performance Evaluation of Twofish Algorithm on IMAN1 Supercomputer

Heba Harahsheh
Department of Computer Science, King Abdullah II School for Information Technology
University of Jordan, Amman, Jordan

Mohammad Qatawneh
Department of Computer Science, King Abdullah II School for Information Technology
University of Jordan, Amman, Jordan

ABSTRACT
Now-a-days data is needed to be exchanged through networks in secure manner. Hence for secure communication required cryptography algorithms. Among these algorithms is Twofish cryptographic algorithm. In this paper the Sequential and parallel implementation have been implemented in IMAN1 supercomputer using Message Passing Interface (MPI). The parallel implementation has been evaluated in terms of execution time, speedup, and efficiency. The simulation results show that 256 key lengths in 4 processors get best efficiency up to 37% while 192 and 128 key length achieves up to 33% and 29% in order.

Keywords
Twofish Algorithm, Parallel, Supercomputer, MPI.

1. INTRODUCTION
Nowadays many systems need huge complex computations in many fields such as industry, and all of these computations use parallel computing in order to get more performance [1]. Parallel and distributed computing systems divide large problems into smaller sub-problems and assign each of them to different processors in a typically distributed system running concurrently in parallel [2] [3] [4] [5] [6].

Transforming data through internet is critical. To secure data transaction and communications, we need to use cryptographic algorithms. Several cryptography algorithms have proposed like Twofish, AES, DES, 3DES, RC2 [7] [8] [9] [10]. Among these algorithms is Twofish cryptography algorithm [11] [12]. Cryptosystems has two types Symmetric Key Encryption (using same key for encryption and decrypting) and Asymmetric Key Encryption (different keys are used for encrypting and decrypting the information). The security of encryption increasing depends on the secure key and strength of cryptographic algorithm.

In this paper, the Performance Evaluation of Twofish algorithm on Supercomputer IMAN1 is presented. IMAN1 is Jordan's first and fastest High Performance Computing resource, funded by JAEC and SESAME [1]. It is available for use by academia and industry in Jordan and the region.

The evaluation is done in terms of the running time, speed and parallel efficiency according to different data size and different number of processors. The results were conducted using IMAN1. It is available for use by academia and industry in Jordan and the region and provides multiple resources and clusters to run and test High Performance Computing (HPC) codes [1].

This paper is organized as follows: Twofish algorithm and related works are presented in Section 2. In section 3, presents the experimental results and comparison. Finally, section 4 concludes the paper.

2. TWOFISH ALGORITHM AND RELATED WORKS
Twofish is a symmetric key block cipher with a block size of 128 bits and key sizes up to 256 bits, and it is related to the earlier block cipher Blowfish [13] [14]. The cipher is a 16-round Feistel network with a bijective F function made up of four key- dependent 8-by-8-bit S-boxes, a fixed 4-by-4 maximum distance separable matrix over GF(28), a pseudo-Hadamard transform, bitwise rotations, and a carefully designed key schedule [15] [16]. Twofish can be easily explained with the diagram as shown in Figure 1; 128-bit plain-text (divided into four parts of 32-bit each) is given for the input whitening where it is XOR-ed with four keys then function g PHT which are explained under the heading twofish functions and modules. Twofish can be implemented in hardware in 14000 gates [10] [17]. The design of both the round function and the key schedule permits a wide variety of tradeoffs between speed, software size, key setup time, gate count, and memory. Twofish has been extensively cryptanalyzed used; even the best attack is able to break only five rounds of the algorithm. Twofish designed with 16-round Feistel network with a bijective F function. The main idea of the round is repeated iteration and generate strong encryption algorithm [10] [18] [19].

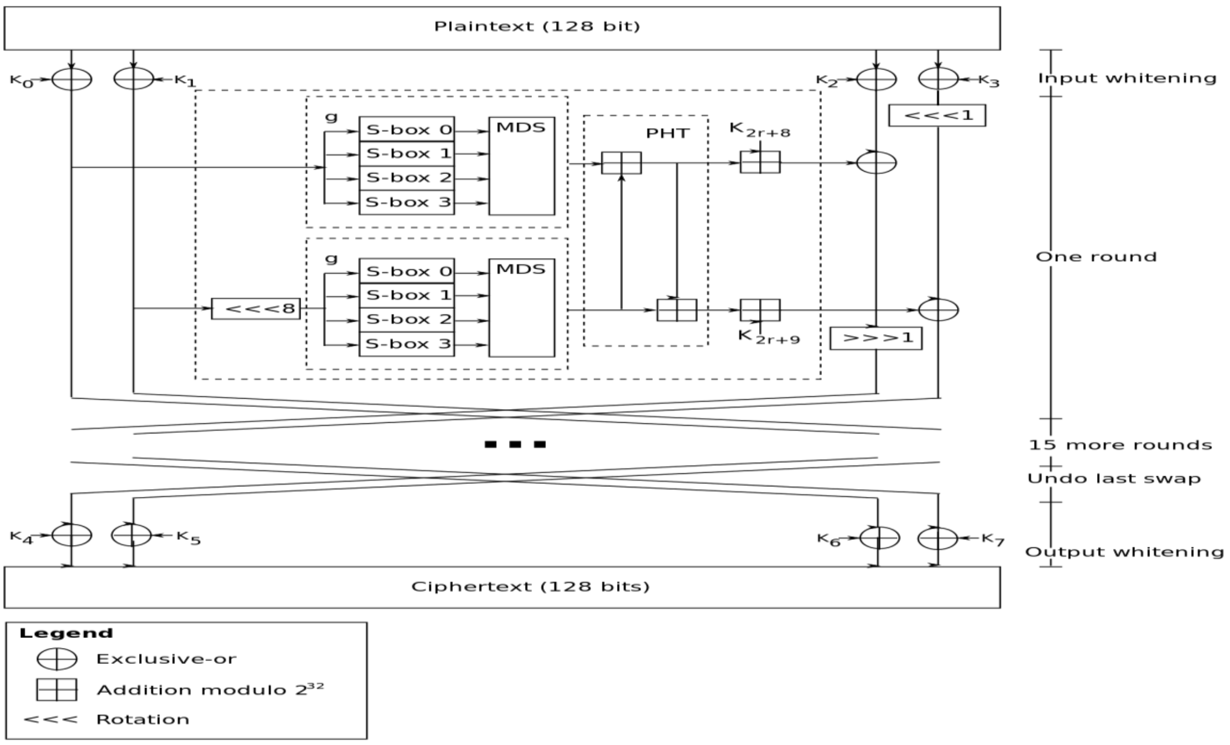

Figure 1: Twofish algorithm structure [20]

Twofish algorithm used in many research areas for getting the best results for securing data. In [2] researcher used agile methods of five phases and implements it using Chilkat library. The data encrypted and decrypted permanently. [1] Extend new cipher algorithm derived from Twofish called Twofish-Ext256. It is same as Twofish algorithm with adding some XOR operations in its design and 10 round Feistel network. The result of comparing between to algorithm shows that Twofish-Ext256 faster than Twofish. [8] The authors used two algorithms Twofish and steganographic algorithm first one for provides the speed of the system and the security, the second algorithm for hiding the encrypted data into an image. The new mixed algorithm coves the security and efficiency.

3. IMPLEMENTATION

A. Environment

Towfish algorithm implemented using C language that widely used in implementation of algorithms. In this paper we use twofish encryption algorithm as a sequential source code and modified it to run in parallel environment using MPI library. We use three key sizes of algorithm 128, 192 and 256 bit for the same text for both executions sequential and parallel code. Sequential test executed on personal computer his specification described in table 1.

Table 1: The Hardware and Software Specifications for Computer used for Sequential execution.

Feature	**Specification**
Processor Clock	Intel(R) Core (TM) i5-3337U CPU.
Memory	4096 MB RAM
Speed	@1.80GHz (4 CPUs).
Operating System	Windows 10 Enterprise 64-bit
File size	35 KB, 80 KB, 260 KB, 550 KB
Number of Processors	1, 2, 4, 8, 16

The same seniors run in a sequential execution also run in parallel code with changing numbers of CPUs used. This experiment done based on different text sizes with three lengths of keys 128, 192 and 256 bits. This testing focuses on the effects of key length size and text size on encryption time. The result of the sequential test compared with the result on a parallel IMAN1 supercomputer.

IMAN1 Zaina cluster user to execute our parallel twofish experiments using open MPI library. Supercomputer hardware and software Specifications listed in table 2.

Table 2: The Hardware and Software Specifications for IMAN1 Zaina cluster

Feature	**Specification**
Processor Clock	Intel(R) Core (TM) I5-6200U CPU
Memory	8.00 GB RAM
Speed	@2.40 GHz
Operating System	Scientific Linux 6.4 with open MPI 1.5.4, C and C++ compiler
File size	35 KB, 80 KB, 260 KB, 550 KB
Number of Processors	1, 2, 4, 8, 16

B. Encryption process in Twofish

Encryption process convert message (plaintext) to unreadable text (ciphertext). The test result using algorithm:

A. Insert plaintext:
Adding message test that went to encrypt in our test input “HI..Jordan First”. That entered to algorithm code.

B. Input key (hexa):

3446464431453537423443394136343338423533373934323536424239 4f4552

C. Execute the algorithm and get the result (chipertext):

60c68d37ed05c881327c2a046bfd3764153c580f60c 68d37ed05c881327c2a046bfd376460c68d37ed05c8

81327c2a046bfd376460c68d37ed05c881327c2a046 bfd3764153c580f064f4f04bb06e4eee3ff211b

C. Decryption process in Twofish

Decryption process returns the chipertext to the original message using the key:

A. Insert chipertext:

60c68d37ed05c881327c2a046bfd3764153c580f60c 68d37ed05c881327c2a046bfd376460c68d37ed05c8 81327c2a046bfd376460c68d37ed05c881327c2a046 bfd3764153c580f064f4f04bb06e4eee3ff211b

B. Input key (hexa):

34464644314535374234433941363433384235333739343235364242394f4552

C. Plaintext : return ciphertext to the original message

“HI..Jordan First”

4. EXPERIMENTAL RESULTS OF TEOFISH ALGORITHM

The parallel twofish algorithm is implemented using open Message Passing Interface (MPI) library, and executed on IMAN1 supercomputer. MPI library provides different functions to support distributing data among different processors to be processed simultaneously. The MPI_Scatterv procedure is used to split and distribute the input data on processors. Every processor executes the same code with the same key for all concurrently on data portion which allocated to it. Parallel twofish evaluated by multiple input sizes (35, 80, 260, and 550) and key sizes of 128, 192, and 256 bits on different number of processors (2, 4, 8, and 16). Figure 3, 4, and 5 shows the execution time of the twofish encryption algorithm on different number of processors with different key lengths.

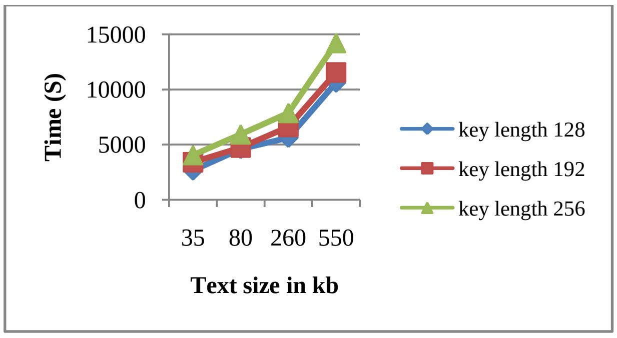

Figure 2: The execution time of twofish algorithm in one processor.

Figure 2 show the execution time of Twofish encryption algorithm on single processor with various packet sizes. The analyses of execution time for execute sequential Twofish implementation on one processor with different plaintext size described in table 1. By comparing execution time to encryption 2670.34(s) for plaintext with size 35 kb using 128 key length with 4056(s) execution time for same text size with 256 key length we see that execution time decreases when we using larger key length. The same behavior appears by increasing the plaintext size as table 1 shows for all experiment plaintext in this case study (35kb, 80 kb, 260 kb and 550kb). In plaintext 550kb the execution time decreases around 3583s when we change key length from 128 to 256. Also when key length changed from 128 to 256 in plaintext size 260 kb the execution time decreases 2246s. According to these results we compared the sequential execution and the parallel execution in this paper for all plaintext size and keys.

Table 3: Experimental analysis of the sequential execution Twofish algorithm with various packet sizes.

key length	**Execution time on 1 CPU with various packet sizes.**			
	35 kb	**80 kb**	**260 kb**	**550 kb**
128	2670.34	4620.11	5631.01	10631
192	3410	4751.2	6620.21	11548.6
256	4056	5936.12	7877.63	14214.9

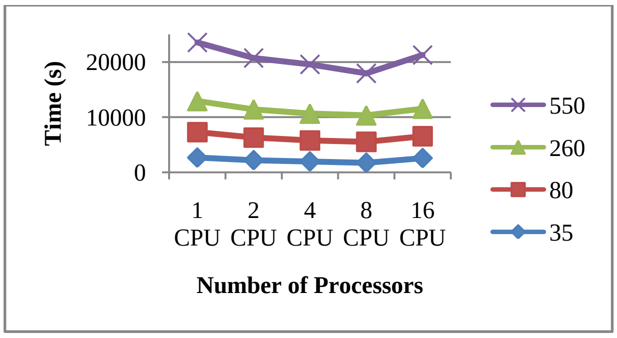

Figure 3: Execution time of twofish - key length 128

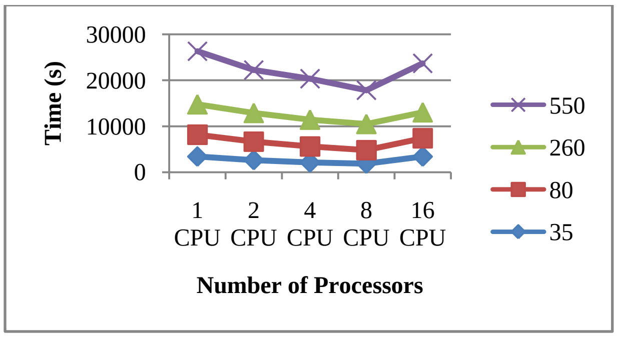

Figure 4: Execution time of twofish - key length 192

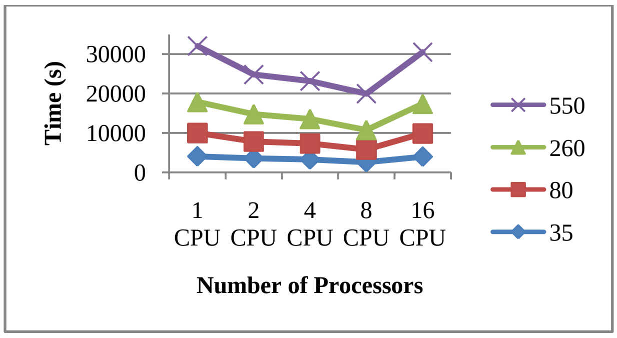

Figure 5: Execution time of twofish - key length 256

In our experiments, we fixed key length size (128, 192, and 256 bits), also fixed file sizes (35, 80, 260 and 550 kb). this is implemented and run in five cases in different numbers of CPUs. For first time the three key sizes run in 2 CPU in variant file size and the experiment repeated with increasing the number of CPUs as the figures 3, 4, 5 shows and Table 3.

The speedup is calculated by taking the ratio between the serial and parallel time. According to the results in Figures 6, 7, 8, 9 the best case of speedup for the data size of 35, 80,260, and 550 KB when the number of processors is equal to 8 and 16, and the speedup decreases when the number of processors is more than 16. Table 5 shows the results for cases in our experiments.

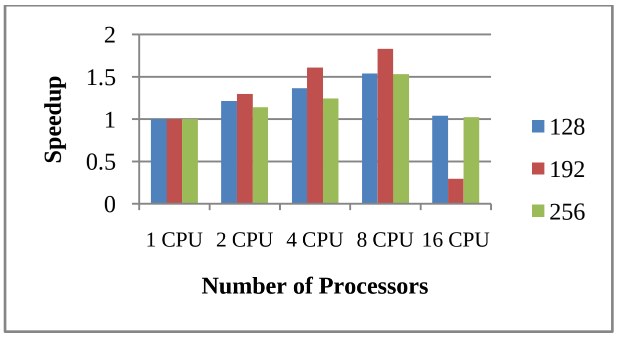

Figure 6: The speedup of the three Key Length and data size 35 kb

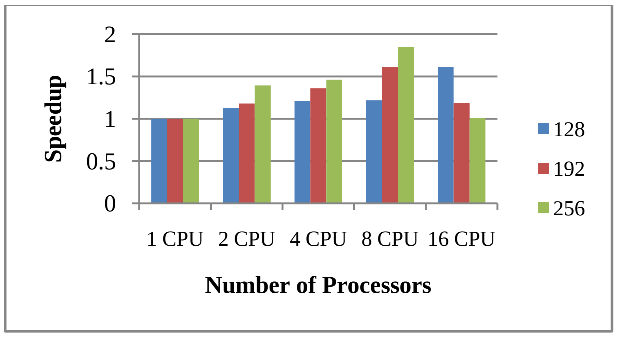

Figure 7: The speedup of the three Key Length and data size 80 kb

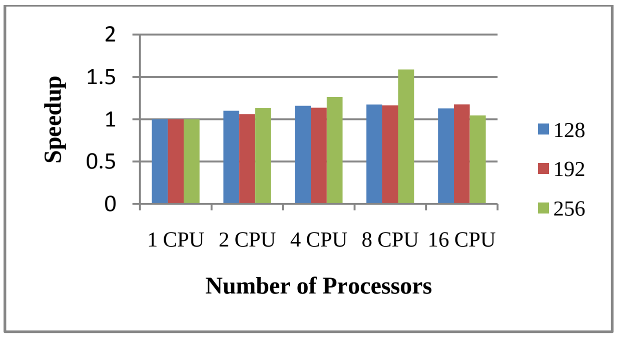

Figure 8: The speedup of the three Key Length and data size 260 kb

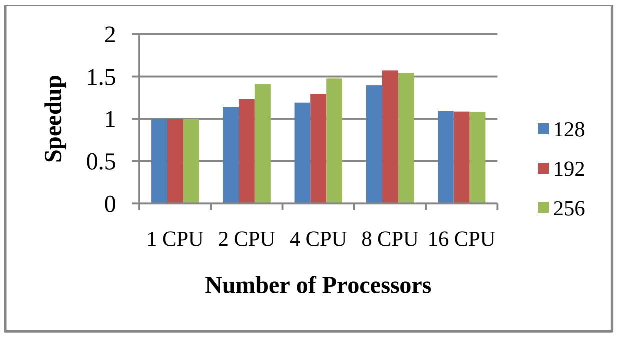

Figure 9: The speedup of the three Key Length and data size 550 kb

Parallel efficiency is computed by taking the ratio between speedup and number of processors. The figures 10, 11, 12, 13 shows the parallel efficiency of twofish algorithm for different packet sizes on different number of CPUs. The results describe that the efficiency is decreased with increasing the key lengths for evaluated packet sizes.

Table 6 present the performance comparisons for runtimes in parallel execution for different numbers of CPUs. The results shows in all cases (in different text size) by increasing number of CPUs the performance decreased because the communication overhead increased; where the benefit of parallelism occurred until 8 CPUs. In 16 CPUs the result is become deceased comparing with 8 CPUs. In text size 550 kb the efficiency is 0.068 for all key lengths. But comparing with 8 CPUs we find the efficiency changed by changing key size.

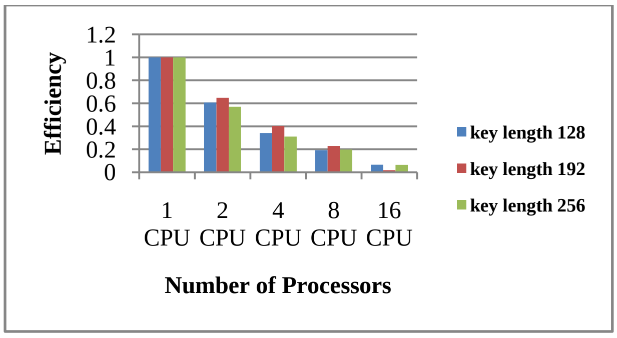

Figure 10: The Efficiency of the Tree Key Length and data size 35 kb

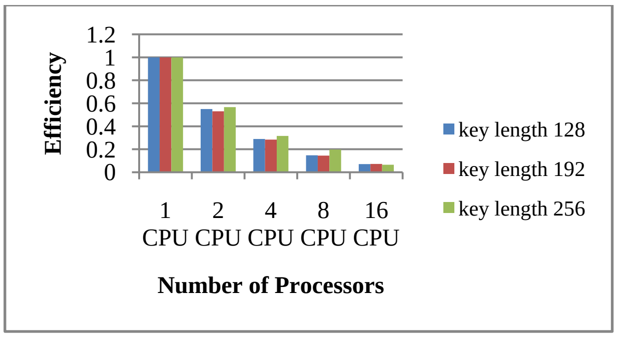

Figure 12: The Efficiency of the Tree Key Length and data size 260 kb

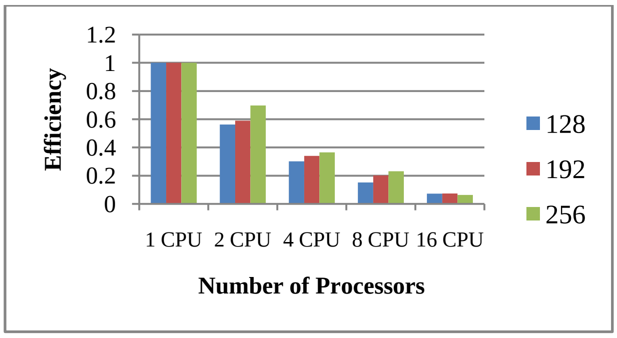

Figure 11: The Efficiency of the Tree Key Length and data size 80 kb

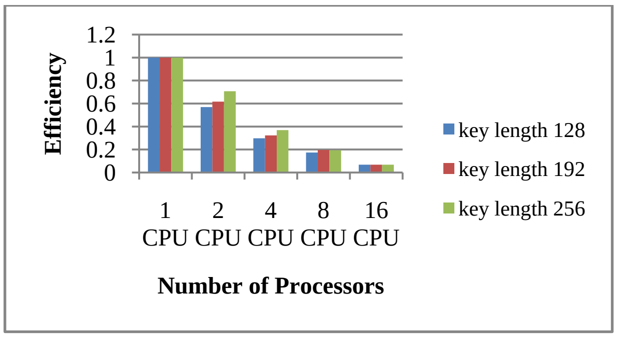

Figure 13: The Efficiency of the Tree Key Length and data size 550 kb

By increasing the CUPs in our experiments and comparing the efficiency of the result at each case. We find that the less of efficiency appears when we start using 16 CPUs, but if we used more than 16 CPU the tests are noteworthy. Table 6 appears the efficiency results for all cases in our experiment in details.

Table 4: comparing results using different key length with different test size and number of processors

Seq.NO	**key length**	**Number of Processors**	**Time (in second) taken on Different text Size in Kb**			
			35	**80**	**260**	**550**
1	128	1	2670.34	4620.11	5631.01	10631.01
2	128	2	2201.3	4102.5	5118	9321.7
3	128	4	1956.04	3826.1	4863.54	8932.8
4	128	8	1736.2	3793.1	4801.2	7621.4
5	128	16	2570.34	3978	4986.35	9745.6
6	192	1	3410	4751.2	6620.21	11548.56
7	192	2	2631	4028.01	6238.04	9354.62
8	192	4	2121	3492.6	5823.7	8925.641
9	192	8	1863	2946.2	5688.3	7352.14
10	192	16	3410	4002	5634.64	10647.25
11	256	1	4056	5936.12	7877.63	14214.86
12	256	2	3562	4259.32	6954.02	10058.33
13	256	4	3263	4063	6235.47	9625.14
14	256	8	2564	3218	4963.41	9217.9
15	256	16	3968	5900	7529.78	13125.78

Table 5: Speedup for Parallel Twofish algorithm with Different File sizes

key length	Speed up of the parallel execution on different number of processors					Text Size(k)
	1 CPU	2 CPU	4 CPU	8 CPU	16 CPU	
128	1	1.213	1.365	1.538	1.039	35
192	1	1.296	1.608	1.83	0.295	35
256	1	1.139	1.243	1.532	1.022	35
128	1	1.126	1.208	1.218	1.61	80
192	1	1.18	1.36	1.613	1.187	80
256	1	1.394	1.461	1.845	1.006	80
128	1	1.1	1.158	1.173	1.129	260
192	1	1.061	1.137	1.164	1.175	260
256	1	1.133	1.263	1.587	1.046	260
128	1	1.14	1.19	1.395	1.091	550
192	1	1.232	1.294	1.571	1.085	550
256	1	1.413	1.477	1.542	1.083	550

Table 6: Efficiency for Parallel Twofish algorithm with Different File sizes

key length	Efficiency for parallel execution on different number of processors					Text Size(k)
	1 CPU	2 CPU	4 CPU	8 CPU	16 CPU	
128	1	0.607	0.341	0.192	0.065	35
192	1	0.648	0.402	0.229	0.018	35
256	1	0.569	0.311	0.198	0.064	35
128	1	0.563	0.302	0.152	0.073	80
192	1	0.59	0.34	0.202	0.074	80
256	1	0.697	0.365	0.231	0.063	80
128	1	0.55	0.289	0.147	0.071	260
192	1	0.531	0.284	0.145	0.073	260
256	1	0.566	0.316	0.198	0.065	260
128	1	0.57	0.298	0.174	0.068	550
192	1	0.617	0.323	0.196	0.068	550
256	1	0.707	0.369	0.193	0.068	550

5. CONCLUSION

In this research, we display the performance of parallel twofish algorithm that was evaluated according to execution time, speedup and efficiency for different sizes of data and various numbers of processors. Parallel twofish in this paper was implemented by C++ using open MPI library and executed on IMAN1 supercomputer. In parallel twofish, According to results, parallel twofish has better execution time for large date size than small data size but with large number of processors on small data size will increase running time instead of decrease execution time because amount of communication between processors will be huge. The experimental results show that the running time will be decreased, and the speed-up of encryption and decryption processes will be increased when the number of processors is 8. Future work for this paper is encryption and decoding of huge measure of content records, pictures, sound records and video files also large text files.

6. REFERENCES

[1] M. Saadeh, H. Saadeh and M. Qatawneh, "Performance Evaluation of Parallel Sorting Algorithms on IMAN1 Supercomputer," 2016.

[2] M. Qatawneh , "embedding linear array network into the tree-hypercube network," 2005.

[3] M. Qatawneh, "Multilayer Hex-Cells: A New Class of Hex-Cell Interconnection Networks for Massively Parallel Systems," 2011.

[4] M. Qatawneh, "Embedding Binary Tree and Bus into Hex-Cell Interconnection Network," 2011.

[5] M. Qatawneh and H. Khattab, "New Routing Algorithm for Hex-Cell Network," 2015.

[6] M. Qatawneh, "New Efficient Algorithm for Mapping Linear Array into Hex-Cell Network," 2016.

[7] Aparna, A. K, J. Solomon, H. M and I. V, "A Study of Twofish Algorithm," 2016.

[8] M. S. Saraireh, "A Novel Security Scheme based on Twofish and Discrete Wavelet Transform," 2017.

[9] M. Ebrahim, S. Khan and U. Bin Khalid, "Symmetric Algorithm Survey: A Comparative Analysis," 2013.

[10] R. KAUR and J. SINGH SAINI, "extended twofish block cipher algorithm for 256-bit blocks and performance comparison with twofish," 2014.

[11] P. Chodowiec, P. Khuon and K. Gaj, "Fast Implementations of Secret-Key Block Ciphers Using Mixed Inner- and Outer-Round Pipelining," 2001.

[12] Mushtaque, H. Dhiman, S. Hussain and S. Maheshwari, "Evaluation of DES, TDES, AES, Blowfish and Two fish Encryption Algorithm: Based on Space Complexity," 2014.

[13] H. K. Verma and R. K. Singh, "Performance Analysis of RC6, Twofish and Rijndael Block Cipher Algorithms," 2012.

[14] P. Gehlot, R. Sharma and S. R. Biradar, "VHDL Implementation of Twofish Algorithm".

[15] D. Rane, "Superiority of Twofish over Blowfish," 2016.

[16] S. Lucks, "The Saturation Attack – A Bait for Twofish," 2002.

[17] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall and N. Ferguson, "On the Twofish Key Schedule," 1999.

[18] M. Dener, "Security Analysis in Wireless Sensor Networks," 2014.

[19] P. Gehlot, S. R. Biradar and B. P. Singh, "Implementation of Modified Twofish Algorithm using 128 and 192-bit keys on VHDL," 2013.

[20] M. A. Devi and M. R. B. S, "Two fish Algorithm Implementation for lab to provide data security with predictive analysis," 2017.