International Journal of Computer Applications (0975 - 8887)
Volume 179 - No.25, March 2018

Performance Improvement of Double Data Encryption Standard Algorithm using Parallel Computation

Amaal Shorman
PhD Student
Department of Computer Science,
King Abdullah II School for Information Technology,
University of Jordan, Amman, Jordan

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

ABSTRACT

Most of the applications require information security on network. Cryptography is a method to provide information confidentiality, authenticity, and integrity. Double Data Encryption Standard algorithm (2DES) is used by several applications to protect their information security. However, current implementations of 2DES algorithm suffer from large execution time. Parallel computation is a promising technique to improve the performance of an algorithm. Divide and conquer method is mainly used in parallel computation to solve the algorithm in parallel by partitioning a task into sub-task and allocating them to available processors. In this paper, sequential and parallel 2DES are evaluated and compared in terms of the running time and the speedup. The parallel 2DES algorithm is implemented using Message Passing Interface (MPI) library, and the results have been conducted using IMAN1 supercomputer. Results showed that the run time of parallel 2DES algorithm outperforms the sequential one. Moreover, on a large number of processors, parallel 2DES achieves better parallel efficiency. Therefore, the parallel 2DES provides much better performance in term of execution time than the serial ones and would be useful to apply it to encrypt and decrypt multimedia.

General Terms

Computer Science, Security

Keywords

Cryptography, Double Data Encryption Standard, MPI, Parallel Computation, Supercomputer

1. INTRODUCTION

Computer networks are required for exchanging information between the sender and the receiver. A key requirements of such networks is ensuring a secure flow of information. Cryptography is one such technique that offers the most secure manner of transmitting sensitive information from the sender to the recipient [11]. A critical objective of a cryptography algorithm is to turn sensitive information indecipherable to everyone except the intended recipient. An example of a cryptography algorithm that offers the much-needed confidentiality is the 2DES algorithm. However, this algorithm suffers from several performance-related drawbacks-for example, execution time and memory requirement [10].

Parallel computation can be deployed for decreasing the execution time of this algorithm [17]. In this case, several computations are conducted at the same time, based on the principle that major problems can usually be split into smaller ones that can be then executed in parallel [5].

The main topic of concern for this paper is scrutinizing the parallel execution of the 2DES algorithm at the data level, considering the nature of this algorithm is not sequential. The MPI Library is utilized for executing the parallel 2DES algorithm. An IMAN1 supercomputer, which engages a distributed memory architecture, is used for carrying out the experiments. This supercomputer is utilized for decreasing the algorithmś execution time. The assessment is carried out with regards to the speedup and execution time as per various file sizes and varying number of processors.

The structure of the remaining paper is as follows: Section 2 offers a synopsis of some related studies. Section 3 offers a synopsis of the DES algorithm. Section 4 comprises a report on the 2DES algorithm. Section 5 talks about the parallelization tools, while the parallelzation of the 2DES algorithm is depicted in the following section. The assessment outcomes of the parallel and sequential 2DES algorithm are outlined in Section 7, while Section 8 presents the conclusion and recommendations for future research.

2. RELATED WORKS

Literature works cite several strategies for augmenting the DES algorithmś performance. Optimization happens either at the software level or the hardware level. The objective of these strategies is to enhance the DES algorithmś efficacy. Rivest [15] provides a straightforward error model for gauging the accuracy of a DES execution in the majority of DES operations. Utilizing a Simple Instruction Multiple Data Stream (SIMD) framework, Biham [2] presents a DES execution that is five times faster on a $64-$bit Alpha computer. This program can also be employed for a quick and exhaustive key search. Anderson et al. [9] applied the DES $S-$box structure to a new block cipher as a candidate for the Advanced Encryption Standard and achieved speed at par with the DES algorithm. Seidel [18] provides a more rapid bit-slice execution of DES devised for the Motorola $G4$ featuring an AltiVec vector process-

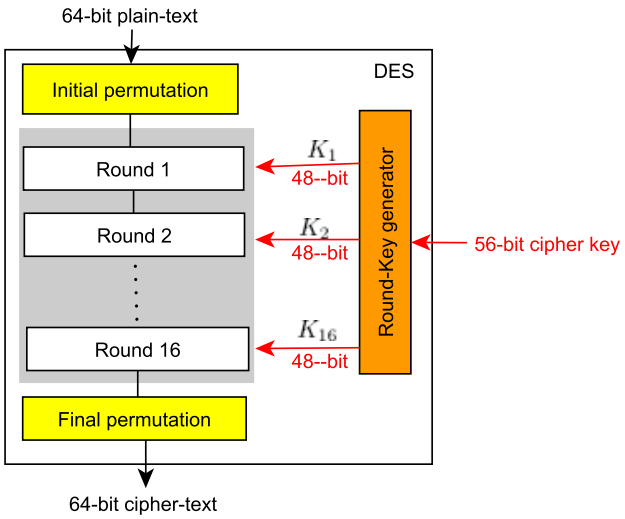


Fig. 1: General structure of DES algorithm.

ing unit. This is an execution running certain tests that are almost nine times quicker than libdes, the most rapid open source DES execution for the *G*4. In [1], Beletskyy et al. depict the outcomes of parallelising DES by deploying OpenMP, an application program interface. They segregated DES algorithm into parallelisable and un-parallelisable segments. The researchers indicate that block ciphers parallelisation in the Electronic Code Book (ECB) mode is conceivable as well as effectual. Celikel et al. [3] perform the design and execution of the DES encryption in the ECB mode with parallelisation schemes based on the following: pipeline, block and plain-text. Parallelisation based on plain-text rendered the best outcomes among the three. Furthermore, the authors indicate that the DES speed is not reliant on the source language. Khaddour et al. [8] investigates the execution of a block cipher Triple Data Encryption Standard (TDES) algorithm on a 16 processor single chip multiprocessor utilizing two parallelisation methods-the first is by data parallelism in which every processor implements the whole TDES algorithm and the second is a pipelined TDES disseminated on the 16 processors. The data parallelism approach offered an improved performance compared to the pipelined technique.

Our design presents a new parallel execution of the 2DES algorithm in the manner that it disseminates the workload among the processors.

3. OVERVIEW OF THE DES ALGORITHM

The DES algorithm is extensively deployed in several applications. As per NIST [17], this algorithm is a symmetric-key block cipher. `Fig.` 1 shows the overall organization of DES cryptography algorithm.
In the course of encryption, DES considers a 64−bit block of plain-text as input and generates a 64-bit block of cipher-text as output by utilizing of permutation as well as substitution. Likewise, at the time of decryption, DES considers a 64−bit block of cipher-text as input and generates a 64−bit plain-text block. The same key is utilized in the course of encryption and decryption, albeit in reverse fashion, wherein the size of the key is 56−bit. There are multiple approaches for the DES algorithm but the ECB approach was used. Here, each 64−bit block of data undergoes encryption individually.

1. Create sixteen sub-keys, each of them is 48 bits long.
   1.1 Choose 64 bit from the key, then remove its parity bits and then make permutation using $(PC-1)$ table.
   1.2 Divide the key into two equivalent parts: the first 28 bits is called C_0 and the last 28 bits is called D_0.
   1.3 Begin with $i=1$, compute the sixteen 48 bit sub-keys K_1-K_16.
      1.3.1 Do left shifts on C_{i-1} and D_{i-1} to obtain C_i and D_i respectively.
      1.3.2 Permute the concatenation $C_i\,D_i$ with $(PC-2)$ table.
      1.3.3 Returned back to 1.3.1 until K_16 has been computed.
2. Make encoding for each 64 bit block of data.
   2.1 Take 64 bit block of plain-text and make permutation using (IP) table.
   2.2 Divide the block into two equal parts: the first 32 bits is called L_0 and the last 32 bits is called R_0.
3. Start with $i=l$, then use sixteen different 48 bits sub-keys to the data block K_1-K_16.
   3.1 Make expansion of 32−bit sub-block R_{i-1} into 48 bits using expansion function (E).
   3.2 Do $\oplus E(R_{i-1})$ with K_i .
   3.3 Divide $E(R_{i-1})\oplus K_i$ into eight 6 bit blocks B_1-B_8.
   3.4 Beginning with $j=l$, substitute the values that are in $S-$boxes for all B, ($S-$Boxes-Substitution Boxes).
      3.4.1 The value of first and last bits B_j, (called m) means the row in $S[j]$ in order to find the value.
      3.4.2 The value of the second and fifth bits B_j, (called n) determines the values in $S[j]$ to find the value.
      3.4.3 Swap B_j, with $S[j][m][n]$.
      3.4.4 Returned back to 3.4.1 until all eight blocks of data have been replaced.
   3.5 Make Permutation of the concatenation of B_1-B_8 with Permutation (P) table.
   3.6 Do $\oplus$ with the value with L_{i-1}. $R_i = L_{i-1}\oplus f(R_{i-1},K_i)$, where $f(R_{i-1},K_i) = P(S[l](Bl)S[l](B1)S[2](B2)S[3](B3)S[4](B4)S[5](B5)S[6](B6)S[7](B7)S[8](B8))$ and where B_j is a six bit block of $E(R_{i-1})\oplus K_i$
   3.7 let $L_i=R_{i-1}$.
   3.8 Returned back to 3.1 until K_16 has been reached.
4. Make permutation of the block $R_16\,L_16$ with the $(IP-1)$. IP and $IP-1$ permutations are inverses with each another.

DES cryptography algorithm uses 16 rounds. `Fig.` 2(a) shows the structure of single round of DES algorithm. On the other hands, `Fig.` 2(b) shows that the DES function is the core of DES algorithm. The DES function takes as input 48 bit of key and produce a 32 bit of output.

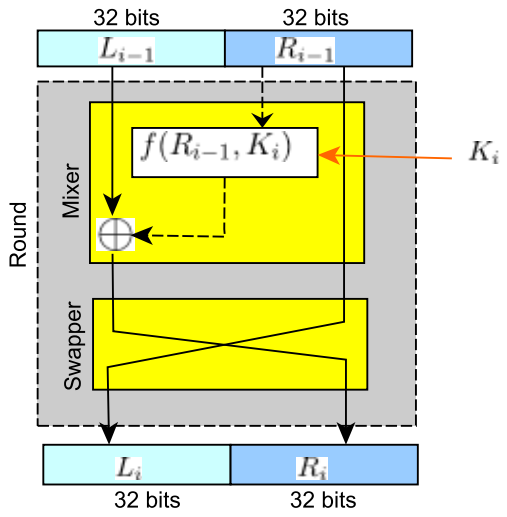
4. DOUBLE DES ALGORITHM

This cipher utilizes instances of DES cipher to encrypt and two instances of reverse ciphers to decrypt. Every instance utilizes a different key and the key size is increased twofold [6]. Considering a plain-text P and two encryption keys K_1 and K_2, a cipher-text can be produced as:

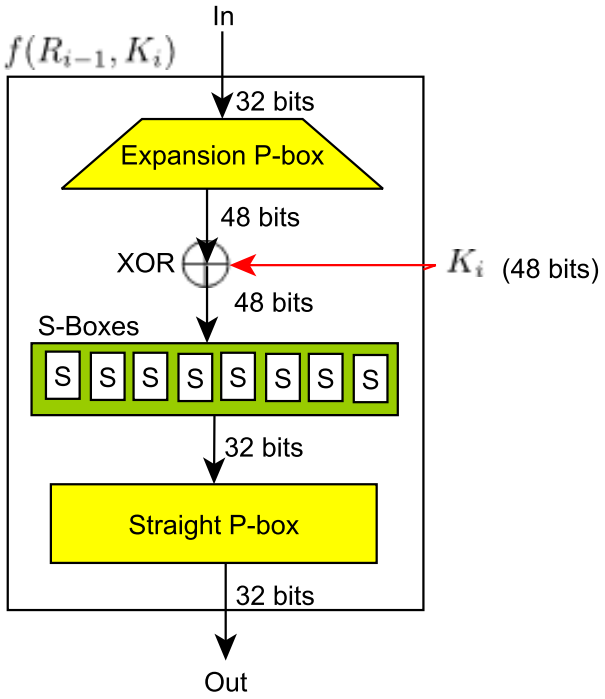
$$C = E(K_2, E(K_1, P))$$

Decryption entails the keys to be employed in reverse manner:

$$P = D(K_1, D(K_2, C))$$



(a)



(b)

Fig. 2: (a) Single Round of DES, (b) DES function.

5. TOOLS OF PARALLELISATION

A. *MPI*

MPI is a homogenous standard of trading messages between several computers that run a parallel program, by employing FORTRAN, C or C++. It is a set of library routines, compiler directives, and environment variables which can be utilized for stipulating shared memory parallelism. MPI is devised for allowing users to formulate programs which can operate competently on the majority of parallel architectures [13].

Furthermore, the MPI can support distributed program execution on a heterogeneous hardware. In other words, one can run a program which begins processes on several computer systems for operating on the same problem. Such processes are unable to communicate with each other by trading information by means of shared memory. Rather, they might employ few of the MPI communication routines. The two fundamental routines are MPI-Send (for sending a message to another process) and MPI-Recv (for receiving a message from another process) [14].

B. *IMAN1 Supercomputer*

The MPI code used the IMAN1 supercomputer. This supercomputer is Jordans first and quickest high-performance computing resource, with funding by SESAME and JAEC. It can be used by the academia and for industrial purposes within the expanse of Jordan. IMAN1 offers several resources and clusters for running and testing high-performance computing codes [12, 16].

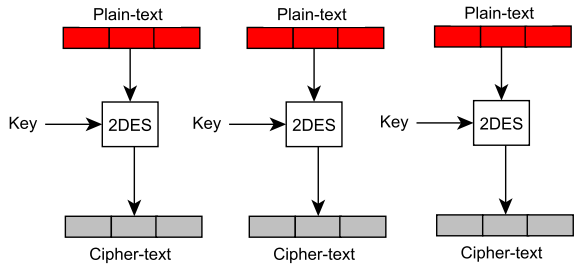


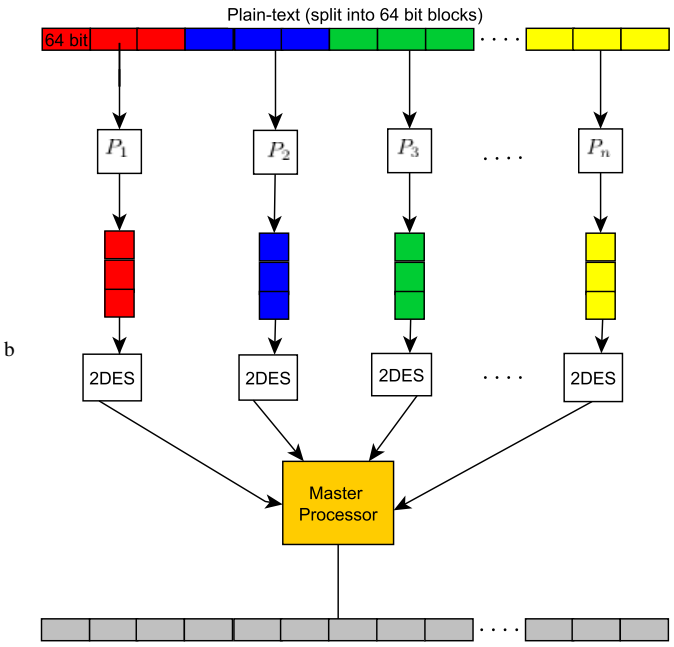
Fig. 3: ECB mode of encryption for 2DES algorithm.

6. PARALLELISATION OF THE 2DES ALGORITHM

The Electronic Code Book (ECB) approach is the easiest way of operation utilised with a block cipher for executing the entire encryption algorithm in cryptography [4]. The whole plain-text is split into chunks of fixed length that can be processed individually. The same key as with the unit encrypts each chunk of plain-text, which is converted into a cipher-text block.

The ECB encryption technique is espoused for executing the 2DES algorithm. `Fig.` 3 depicts the ECB encryption technique for this algorithm. Blocks 64−bit in length are generated from the mentioned plain-text. The ECB mode aids a parallel architecture since the individual plain-text blocks can be processed separately. The cipher-textś decryption can be carried out in a similar manner.
The sequential algorithm could be rendered certain modifications so as to capitalise on the multiprocessing units. As per the parallel computation paradigms [7], the individual parts of the algorithms should be determined and then equipped to work in an independent processor.

2DES is relatively sequential in nature because every following round is dependent on the output of the previous round. Therefore, there is no attention to accelerate the 2DES encryption itself,



b

Fig. 4: Block diagram for Parallel Implementation of 2DES Algorithm.

Table 1. : The Hardware and Software Specifications.

Hardware specification	Dual Quad Core Intel Xeon CPU with SMP, 16 GB RAM
Software Specification	Scientific Linux 6.4 with open MPI 1.5.4, C and C++ compiler.
File size	8KB, 16KB, 32KB,64KB,128KB, 265KB, 1MB
Number of Processors	1, 2, 4, 8, 16, 32, 64

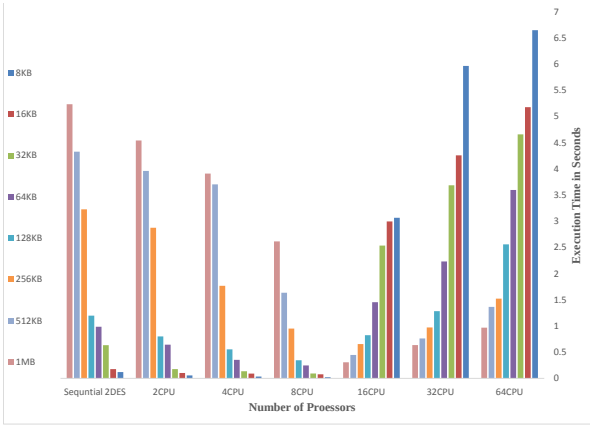


Fig. 5: Execution Time for Sequential and parallel 2DES.

but instead the blocks are encrypted separately. Doing this successfully would offer us significant gains in efficacy as well as speedup. Fig. 4 depicts the steps that are carried out to do the parallelisation according to the steps given below:

- Allot every processor a copy of the whole data;
- Every processor is allocated a part of the data, divided into 64−bit blocks;
- Every block then undergoes encryption by a processor for creating cipher-text blocks; every processor encrypts its blocks separately; however, the blocks themselves are encrypted in sequence per processor;
- Data is reclaimed using a master processor; cipher-text is written to the output file.

The result of the parallelisation process is a parallelised 2DES algorithm that depicts enhanced performance over a sequential execution.

7. PERFORMANCE ASSESSMENT AND RESULTS

The execution results mentioned in this section make a comparison between the parallel and sequential execution of the 2DES algorithm. The sequential 2DES was executed utilizing C programming language. It has been assessed with respect to the execution time of encryption as well as decryption. The sequential experiment was carried out on a laptop with an Intel(R) Core(TM) $i7-4500U$ having a RAM of 8GB and a speed of 2.4 GHz. The file sizes were in the range of 8KB to 16MB.

Conversely, the parallel 2DES was executed using MPI library and C language. It was assessed with respect to speedup, running time, and parallel efficacy performance metrics as per various file sizes and varying quantity of processors. On average, three runs are considered for recording the outcomes. The parallel experiment was carried out on a IMAN1 supercomputer real distributed system. The software and hardware details coupled with the used execution parameters are mentioned in Table 1.

7.1 Run Time Assessment

Table 2 depicts the run time of parallel and sequential execution of the 2DES algorithm for various file sizes with varying quantity of processors. As shown in the table, as the size of data grows, the run time goes up because of the higher time needed for data splitting and collection. The outcomes indicate that deploying up to 8 processors at the same time with the 2DES can deliver improved outcomes compared to the sequential one. It can be concluded that as the size of the file grew, the parallelised code with two processors decreased the time in comparison with the sequential time required for encrypting a file. Fig. 5 indicates that the sequential codes execution time is more compared to the parallelised code. When the size of the file is under 16KB, the difference is negligible. Thus, when the size of the file grows, the differences between parallel and sequential running time would rise considerably.

7.2 Speedup Assessment

The 2DES is also assessed with regards to speedup i.e., the ratio of the sequential time to the parallel time. Fig. 6 depicts the execution time on 2, 4, 8, 16, 32, and 64 processors with varying file sizes. Furthermore, it indicates that employing up to 8 processors can deliver the best outcomes in terms of encryption for all cases. For 16, 32, and 64 processors, the running time rose as the size of the file grew, driven by the communication between the processors to send and obtain data. As the size of the file and the quantity of processors rose, the communication between the processors was augmented as well; this took more time compared to the time required for sequential. It can be concluded that irrespective of the available

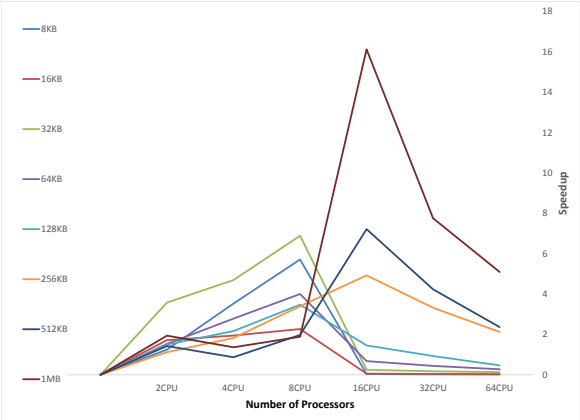


Fig. 6: The Speedup of parallel 2DES.

8. CONCLUSION AND FUTURE WORKS

2DES is utilized in several real-time applications. Therefore, it is essential to encrypt and decrypt big files in real time. The objective of this paper is to decrease the time of encryption and decryption for the 2DES algorithm. A parallel 2DES encryption algorithm in ECB operation mode is devised and executed by utilizing the MPI library. Here, the plain-text is split into equal partitions which equal the quantity of processors available. The experiment outcomes are executed on an IMAN1 supercomputer. Following the assessment of the execution time and the speedup, the parallel execution of the 2DES algorithm utilizing more than one processor consumes less time and offers improved speedup for encrypting and decrypting large-sized files, like videos, compared to the sequential execution. In general, one can say that parallel computation offers an effectual and dependable mode of executing the 2DES algorithm.

Future research will encompass effectual parallel executions of other encryption algorithms, assessed using the IMAN1 supercomputer for decreasing their execution time.

Acknowledgment

We would like to thank Eng. Zaid Abudayyeh for his support to accomplish this work.

9. REFERENCES

[1] V Beletskyy and D Burak. Parallelization of the data encryption standard (des) algorithm. In *Enhanced methods in computer security, biometric and artificial intelligence systems*, pages 23–33. Springer, 2005.

[2] Eli Biham. A fast new des implementation in software. In *International Workshop on Fast Software Encryption*, pages 260–272. Springer, 1997.

[3] Ebru Celikel, Jean Davidson, and Colin Kern. Parallel performance of des in ecb mode. In *Computer Networks, 2006 International Symposium on*, pages 134–139. IEEE, 2006.

[4] Morris J Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. 2001.

[5] Ananth Grama. *Introduction to parallel computing*. Pearson Education, 2003.

[6] Atul Kahate. *Cryptography and network security*. Tata McGraw-Hill Education, 2013.

[7] Hironori Kasahara and Seinosuke Narita. Practical multiprocessor scheduling algorithms for efficient parallel processing. *IEEE Trans. Comput.;(United States)*, 100, 1984.

[8] Mazen KHADDOUR, Zhoukun Wang, and Omar Hammami. Performance evaluation and analysis of parallel software implementations of tdes on a16-peembedded multiprocessor platform. In *Network and Service Security, 2009. N2S'09. International Conference on*, pages 1–5. IEEE, 2009.

[9] Ross Anderson1 Eli Biham2 Lars Knudsen. Serpent: A proposal for the advanced encryption standard. In *First Advanced Encryption Standard (AES) Conference, Ventura, CA*, 1998.

[10] Wei Liu, Rong Luo, and Huazhong Yang. Cryptography overhead evaluation and analysis for wireless sensor networks. In *Communications and Mobile Computing, 2009. CMC'09. WRI International Conference on*, volume 3, pages 496–501. IEEE, 2009.

[11] Wei Liu, Beihua Ying, Huazhong Yang, and Hui Wang. Accurate modeling for predicting cryptography overheads on wireless sensor nodes. In *Proceedings of the 11th international conference on Advanced Communication Technology-Volume 2*, pages 997–1001. IEEE Press, 2009.

[12] IMAN1 Project. Jordan's national supercompting center. `http://www.iman1.jo/iman1/`. Accessed: 2017-11-25.

[13] J Quinn Michael. Parallel programming in c with mpi and openmp mcgraw-hill inc. Technical report, ISBN 0-07-058201-7, 2004.

[14] Rolf Rabenseifner, Georg Hager, and Gabriele Jost. Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In *Parallel, Distributed and Network-based Processing, 2009 17th Euromicro International Conference on*, pages 427–436. IEEE, 2009.

[15] Ronald L Rivest. *Testing implementations of DES*. MIT Laboratory for Computer Science, Cambridge, Mass, 1985.

[16] Maha Saadeh, Huda Saadeh, and Mohammad Qatawneh. Performance evaluation of parallel sorting algorithms on iman1 supercomputer. *International Journal of Advanced Science and Technology*, 95:57–72, 2016.

[17] Bruce Schneier. *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, 2007.

[18] E Seidel and Joseph N Gregg. Preparing tomorrows cryptography: Parallel computation via multiple processors, vector processing, and multi-cored chips. *Senior Honors Project, Lawrence University*, 2003.

Table 2. : Execution Times of Sequential and Parallelized Code in Seconds.

File Size	Execution Time						
	Sequential Implementation	Parallel Implementation					
		Number of Processors					
		2	4	8	16	32	64
8KB	0.120	0.056	0.034	0.021	3.067	5.966	6.65
16KB	0.177	0.103	0.091	0.078	2.996	4.259	5.178
32KB	0.634	0.177	0.135	0.092	2.536	3.687	4.66
64KB	0.985	0.643	0.354	0.246	1.455	2.233	3.597
128KB	1.198	0.801	0.553	0.345	0.824	1.284	2.559
256KB	3.228	2.876	1.768	0.95	0.655	0.972	1.522
512KB	4.328	3.961	3.704	1.635	0.447	0.762	1.365
1MB	5.235	4.543	3.91	2.615	0.306	0.636	0.969