Contemporary Engineering Sciences, Vol. 8, 2015, no. 18, 809 - 816
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2015.58230

Possibilities for Steganographic Parallel Processing with a Cluster System

Stanimir Zhelezov and Hristo Paraskevov

Department of Computer Systems and Technologies
Faculty of Mathematics and Informatics
Konstantin Preslavski University of Shumen, 9712 Shumen, Bulgaria

Abstract

The paper suggests a parallel algorithm for the implementation of the LSB steganographic method. A program for cluster computer system in OpenMPI environment is implemented and an assessment of the acceleration of the program depending on the number of the computing cores that implement the program is made.

Keywords: Steganography, Steg analysis, Parallel processing, Cluster systems

Introduction

The analysis of a large amount of suspicious stego files requires vast computational resources, making it necessary to seek ways for accelerating this process by the methods of parallel processing [3], [4]. Data for parallel algorithms with cluster systems is accessible [1], [3], but for understandable reasons there lacks sufficient information on the use of parallel processing for the needs of steganography. The proposed method could be used as a main part of a parallel information extraction in [2], [5] and [6]. Another parallel cryptographic algorithm is presented in [7].

The experiments carried out at the *Computer Security* research laboratory at the Faculty of Mathematics and Informatics of Konstantin Preslavski University of Shumen with steganographic programs accessible from the Internet (S-tool, Invisible Secrets 4, SecurEngine and Camouflage) with a consistent computer showed big differences in the execution time of the programs depending on the

size of the container and the message - from several seconds to several minutes for one stego file, at that the efficiency of the insertion of the message is also different. Bigger differences are obtained with the steganalysis of a suspicious stego file.

Steganography Algorithm

The most popular steganographic algorithm is the LSB method (Least Significant Bit) [8]. Therefore, it was logical to begin with the development of parallel steganographic programs with the application of this approach.

The current work suggests a steganographic algorithm with the purpose to increase the speed for hiding information in a *.BMP file without damaging both the information and the *.BMP container. It has a high degree of usefulness with minimal changes in the graphic representation, invisible to the human eye. The only limitation is to the amount of input information which cannot be greater than one-eighth of the size of the container.

In the basis of the proposed algorithm lies the idea of parallel processing of certain parts of the message and the container as follows:

1. The secret message is divided into fragments - the size of each fragment is determined by the number of cores of the cluster involved in the implementation of the program.

2. The container is divided into segments - the size of each segment is determined by the size of the fragment of the message (the number of the bytes of the fragment) multiplied by 8.

3. Each core processes a fragment of the message and inserts it in an appropriate segment of the container.

4. The result of processing the *i*-th core (fragment *i* and segment *i*) is the formation of a block *i* of the stego file.

5. The blocks obtained from each core are combined into a common stego file. The combining of the blocks is carried out by the controlling console computer of the cluster system.

The implementation of the algorithm starts with an initial defining of the size of the container file in bytes.

Figure 1 shows the block scheme of the developed parallel algorithm for realization of the steganographic LSB method.

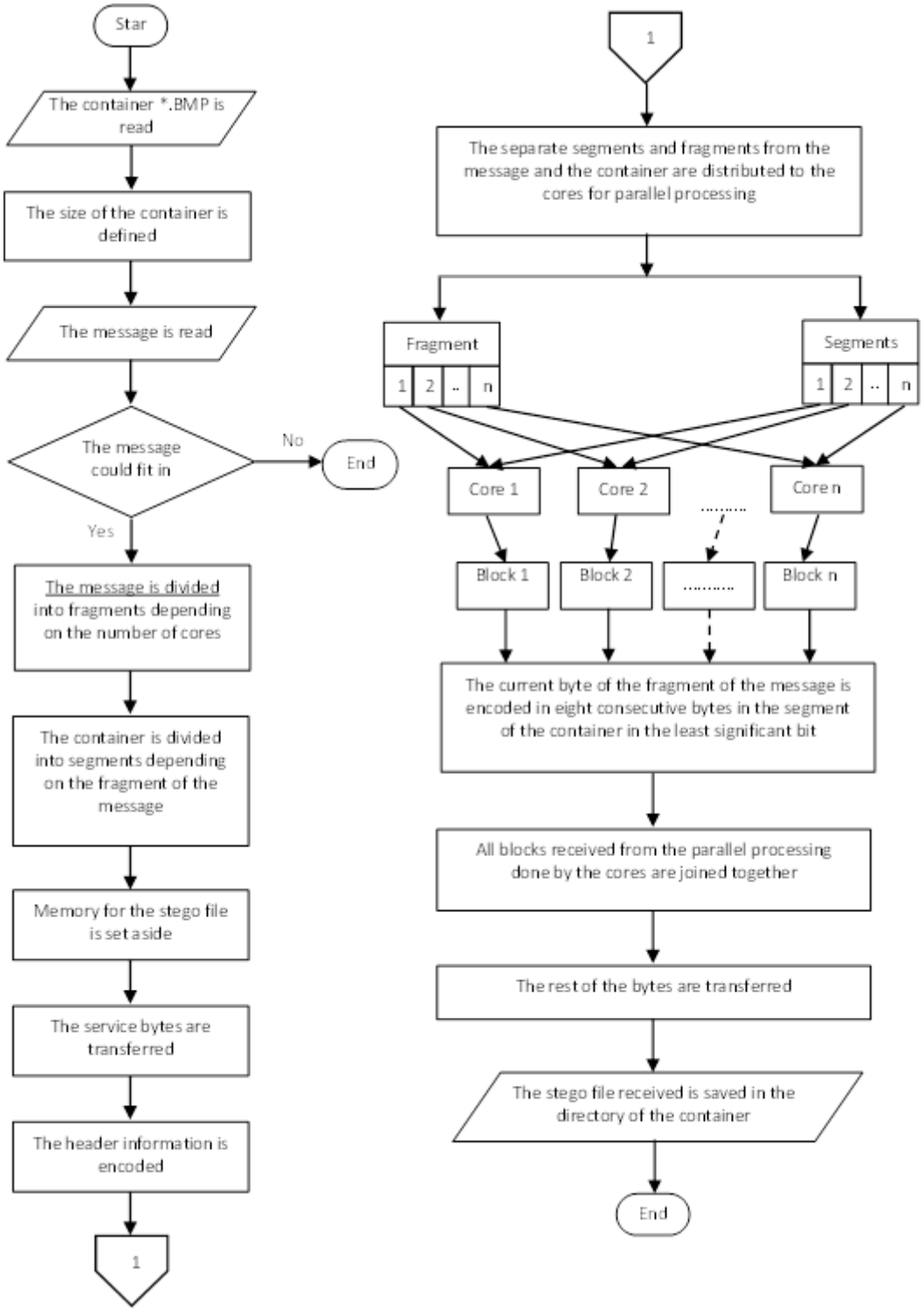

Figure 1. Block diagram of the proposed parallel algorithm for realization of the steganographic LSB method with the Radian cluster computer system

Program Realization and Tests Results

For the purposes of the realization of the developed algorithm a working parallel program „ParStego v1.1" has been developed for encoding steganographic information in a *.BMP file on a Radian cluster computer system.

The architecture of the Radian cluster system is classified as highly connected homogeneous computing cluster with distributed memory. It consists of a control console computer and Ethernet nodes that are linked with it in a network. The cluster nodes are 8 – these are standard system units of personal computers with normal operating temperature range. Each of them has four core processor Intel Core2 Q 8300 at 2,5 GHZ, 1 GB of operational memory and hard drive capacity 160 GB. The total theoretical performance of the developed system is 126 Gflops under Intel LINPACK64. Since the algorithm lacks a block for preliminary compression, the maximum amount of information that will be embedded in the image is determined by the size of the container file minus the header information divided by eight. The size of the stego file must be equal to that of the container file.

The program is realized using the C++ language with OpenMPI and experiments have been carried out with it under the following scheme.

First, a database is created of containers with *.BMP format that are used during the experiments. After that, messages with *.TXT format are embedded in them. The messages should not exceed one-eighth the size of the image (this results from the LSB method).

Tables 1 and 2 show the column of the container and the volume of the file that is to be hidden. For all the conducted experiments the time for implementing the program from 1 to 18 cores was tested.

The following symbols are used in the tables:

Vk - Volume of the container in bytes. The row is divided into two parts, respectively:

Size – The size of the container where the message will be embedded;

Segment - The volume of the fragment in bytes is multiplied by 8 to obtain the so called segment of the container file.

Vmessage – Volume of the message in bytes. The row is divided into two parts, respectively:

Size – The size of the message that is to be embedded;

Fragment – The message that should be delivered is divided into fragments. Their number is determined by the ratio of the volume of the message to the number of cores involved in the parallel processing.

Number of processes – The number of the cores on which the program runs.

Timp. – Time to implement the program on all cores.

Tpr. – The relative time for implementation of the program based on the number of processes.

Number of processes	V_c (bytes)	Segment (bytes)	$V_{message}$ (bytes)	Fragment (bytes)	T_{impl}
1	3145782	3145782	393206	393206	0,90829
2	3145782	1572824	393206	196603	0,85829
3	3145782	1048544	393206	131068	0,80839
4	3145782	786408	393206	98301	0,75849
5	3145782	629128	393206	78641	0,70859
6	3145782	524272	393206	65534	0,65869
7	3145782	449376	393206	56172	0,60879
8	3145782	393200	393206	49150	0,55889
9	3145782	349512	393206	43689	0,50899
10	3145782	314560	393206	39320	0,45909
11	3145782	285968	393206	35746	0,40919
12	3145782	262136	393206	32767	0,35929
13	3145782	241968	393206	30246	0,30939
14	3145782	224688	393206	28086	0,25949
15	3145782	209704	393206	26213	0,20959
16	3145782	196600	393206	24575	0,15969
17	3145782	185032	393206	23129	0,10979
18	3145782	174752	393206	21844	0,05989

Table 1. Results from the carrying out of the tests with size of the file 3145782 bytes

Number of processes	V_c (bytes)	Segment (bytes)	$V_{message}$ (bytes)	Fragment (bytes)	T_{impl}
1	9437238	9437104	1179638	1179638	2,89927
2	9437238	4718552	1179638	589819	2,73937
3	9437238	3145696	1179638	393212	2,57947
4	9437238	2359272	1179638	294909	2,41957
5	9437238	1887416	1179638	235927	2,25967
6	9437238	1572848	1179638	196606	2,09977
7	9437238	1348152	1179638	168519	1,93987
8	9437238	1179632	1179638	147454	1,77997
9	9437238	1048560	1179638	131070	1,62007
10	9437238	943704	1179638	117963	1,46017
11	9437238	857912	1179638	107239	1,30027
12	9437238	786424	1179638	98303	1,14037
13	9437238	725928	1179638	90741	0,98047
14	9437238	674072	1179638	84259	0,82057
15	9437238	629136	1179638	78642	0,66067
16	9437238	589816	1179638	73727	0,50077
17	9437238	555120	1179638	69390	0,34087
18	9437238	524280	1179638	65535	0,18097

Table 2. Results from the carrying out of the tests with size of the file 9437238 bytes

For all experiments carried out, the execution time of the program with number of the cores from 1 to 18 was tested. Based on the data of the results obtained Figure 2 was developed.

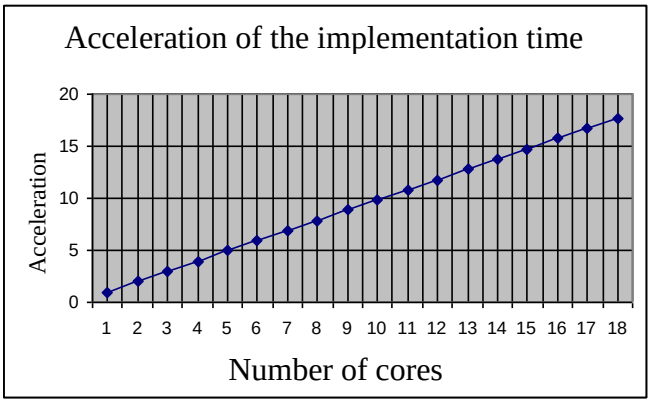

Figure 2. Acceleration of the implementation time depending on the number cores used

The effectiveness of the program has been tested in the case when each process-worker uses as many threads as the number of processor cores available on the machine where it is being executed.

Figure 3 shows the screens when implementing the program with one and ten cores. It contains the following information:

– Starting number of cores: 10 – Number of the cores under which the tasks necessary for the implementation of the algorithm will be allocated;

– Information for container file – The information about the container file is displayed here. It consists of Image size – the size of the file and Image chunk – the size of the segment in bytes;

– Information for text file – The information about the message is displayed here. It consists of Text length – the size of the message and Text chink – the size of the fragment in bytes;

– Time to send data to remote hosts – The time for sending data to remote hosts in seconds;

– Time for work on core 0 ÷ 9 – Time for work on the separate cores in seconds;

– Time to work on all cores – Time to work on all cores in seconds;

Total time with transfer of data – The total time for work with data transferring.

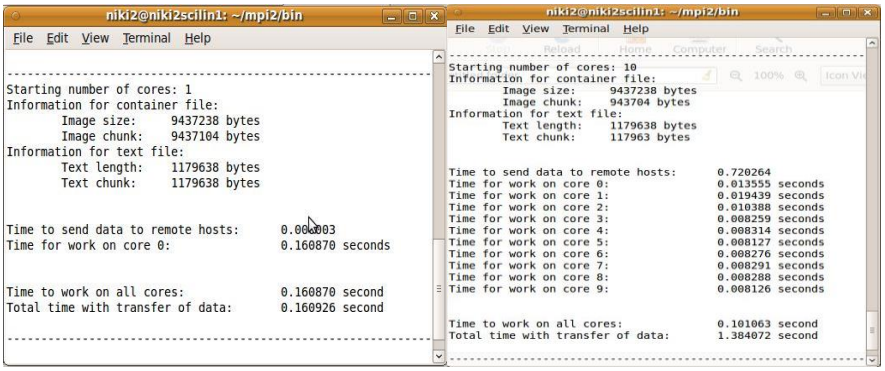

Figure 3. The screens when implementing the program with one and ten cores

From the experiments carried out the following conclusions could be drawn:

The execution time of the program sharply decreases depending on the number of the cores used. From the analysis of the network connections of the nodes in the cluster system it has been determined that the time needed for transferring data between the nodes is highly dependent on the hardware realization of the network, so it has been ignored.

The result of the parallel realization with n = 18 cores shows uniform load of the cores used and decrease of the time to hide the information up to 16 times, i.e. n-2 times in comparison with the sequential processing of the same program.

The parallel algorithm with realization of the LSB is only a step in the utilization of the methods for realization of parallel programs of the Radian cluster system at the University of Shumen. No means of pre-compression and/ or encryption are applied in it. This is the subject of further development of the algorithm, which will lead to an increase in the volume of the hidden message, while maintaining the ratio of the embedding 0.12, depending on the LSB method. On the basis of the tests carried out with the program there could be searched criteria for detecting messages hidden by this method.

After the realization of the parallel algorithm for hiding LSB one can proceed to researching and developing own parallel algorithm for steganalysis.

It can be expected that the application of parallel programs for steganalysis with the Radian cluster system will accelerate the discovery of hidden messages from 2 to 16 times, compared to the sequential processing of the same data with the same type of processors.

Acknowledgements. This paper is supported by the Project RD-10-686 / 01.04.2015 “Steganography and security in cloud systems”. The Project is realized by the financial support of the Konstantin Preslavski University of Shumen, Bulgaria.

References

[1] I.S. Dhillon, D.S. Modha, A Data-Clustering Algorithm on Distributed Memory Multiprocessors, *Large-Scale Parallel Data Mining, Lecture Notes in Computer Science*, **1759** (2000), 245 - 260. http://dx.doi.org/10.1007/3-540-46502-2_13

[2] A. Nachev, M. Hogan, B. Stoyanov, Cascade-Correlation Neural Networks for Breast Cancer Diagnosis, In ICAI'11: *The 11th International Conference on Artificial Intelligence*, 475 - 480.

[3] C.F. Olson, Parallel algorithms for hierarchical clustering, *Parallel Computing*, **21 (**1995), 1313 - 1325. http://dx.doi.org/10.1016/0167-8191(95)00017-i

[4] A. Plaza, D. Valencia, J. Plaza, P. Martinez, Commodity cluster-based parallel processing of hyperspectral imagery, *Journal of Parallel and Distributed Computing*, **66** (2006), 345 - 358. http://dx.doi.org/10.1016/j.jpdc.2005.10.001

[5] T. Stefanov, System for Information Extraction from News Sites, *Mathematical and Software Engineering*, **1** (2015), 25-30.

[6] B.P. Stoyanov, Chaotic cryptographic scheme and its randomness evaluation, in 4th AMiTaNS'12, AIP CP, **1487** (2012), 397 - 404. http://dx.doi.org/10.1063/1.4758983

[7] B. Stoyanov, K. Kordov, Pseudorandom Bit Generator with Parallel Implementation, In Large-Scale Scientific Computing 2013, *Lecture Notes in Computer Science*, **8353** (2014), 557 - 564. http://dx.doi.org/10.1007/978-3-662-43880-0_64

[8] Da-Chun Wua, Wen-Hsiang Tsai, A Steganographic Method for Iimages by Pixel-Value Differencing, *Pattern Recognition Letters*, **24** (2003), 1613 - 1626. http://dx.doi.org/10.1016/s0167-8655(02)00402-6

Received: August 23, 2015; Published: September 3, 2015